Chapter 7
Defend the Computing Environment

Defense of the computing environment focuses on the use of information assurance (IA) technologies to ensure the availability, integrity, and privacy of user information as it enters, leaves, or resides on clients and servers. Clients are the end-user workstations, both desktop and laptop, including peripheral devices, whereas servers include application, network, Web, file, and communication servers. Applications running on clients and servers may include secure mail and Web browsing, file transfer, database, virus, auditing, and host-based intrusion detection systems (IDS) applications. Defending the computing hardware and software from attack may be the first line of defense against the malicious insider—or it may be the last line of defense against the outsider who penetrates the enclave boundary defenses. In either case, defending the computing environment is necessary to establish an adequate IA posture.

As illustrated in Figure 7-1, the computing environment may reside within a physically protected enclave, or it may be the host platform of a traveling user. The environment includes the host or server applications, operating system (OS), and client/server hardware. To date, the Defense-in-Depth technology strategy has identified the need for secure applications and OS on clients and servers. These security technologies are addressed in Section 7.1, Security for System Applications. The secure applications considered are secure messaging, secure Web browsing, file protection, and mission-specific applications. Virus and intrusion detection software installed on host platforms is covered in Chapter 6, Defend the Enclave Boundary/External Connections.

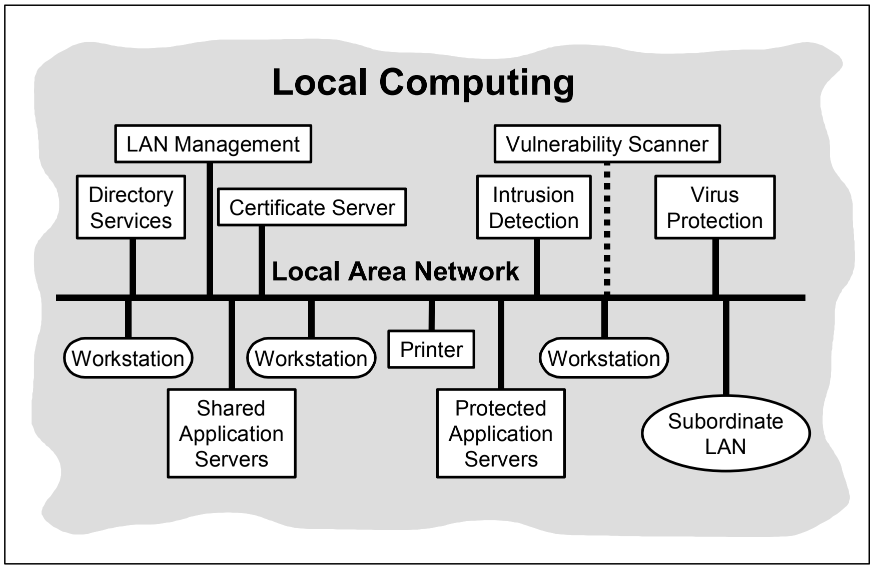

Figure 7-1. Local Computing Environments

Security-Enabled Applications. An application is any software written to run on a host; it may include portions of the OS. Although there are multiple strategies for security-enabled applications, this framework emphasizes the use of open standards and commercial off-the-shelf (COTS) solutions. The evolution of application programming interfaces (API) will simplify and improve the interoperability of the solutions and produce standards for use throughout the Government and the commercial community.

Securable Operating System. In general, the IA strategy is to provide a centrally managed, securable, and securely configured operating system foundation. The vast majority of a system's life occurs after it is initially configured. System administrators should employ tools to ensure that the initial configuration is secure, that only needed services are enabled, that vendor updates and patches are maintained, that subsequent changes maintain or improve security configuration, and that systems are checked regularly to ensure that the configuration remains secure.

Host-Based Monitoring. Host-based monitoring technologies include detection and eradication of malicious software like viruses; detection of software changes; checking of configuration changes; and audit, audit reduction, and audit report generation. Monitoring mechanisms include tools run by users, such as antivirus software, and tools managed by system administrators. For example, administrators use network and host-based vulnerability analysis tools to verify that vendor patches are installed, detect weak user passwords, and monitor for excessive use of user access privileges. Virus protection software should be used within local computing environments.

7.1 Security for System Applications

This section examines the security features and services that applications can or should provide, particularly with respect to the use of cryptography and good design practices. Several technology areas are considered:

- Network-to-network communication.
- Cryptographic security services and cryptographic application programming interfaces (CAPI) that provide generic encryption, key exchange, signature, and hash functions, or higher level security services for application developers.
- Executable content or software download, including software upgrade issues, e.g., firmware updates.
- Applications themselves that can be basic and relatively straightforward, taking advantage of security services for their functionality, or extremely complex, adapting basic functionality to meet a particular mission need.

In each technology area, the section describes specific security considerations and security and interoperability concerns. These include alternative technologies, protocols, and standards for interoperability that may be useful to those building complete and real systems.

This section generally follows the format established in other sections: target environment, consolidated requirements, potential attacks, potential countermeasures, technology assessment, cases, and guidance. The concerns for e-mail, distributed databases, file encryption, Internet phone, and Web-based applications have similarities but also differences because of use, technology, and standards. In the major sections, the common aspects of application-level security are considered. In the technology assessment section, additional, more application-specific information is supplied.

7.1.1 Target Environment

The environment for user- or application-layer security is generally considered to be a workstation (laptop, desktop, etc.) connected at least part of the time via a network to sensitive information servers. Additionally, the information on the servers (and on the workstation) may need protection even from other personnel or workstations privileged with access to network resources. Further, the section assumes that the environment is the “application space” where users and applications operate on information that has value. Physically, this environment applies anywhere within the Global Information Infrastructure (GII) that a particular application might send, store, retrieve, or destroy sensitive information. It typically embodies the elements of a three-tier model: the client, the business process, and the databases that serve a particular process.

7.1.1.1 Applications Environment

The environment for applications is considered to be a well-managed UNIX or Windows-based client/server OS, managed by knowledgeable system administrators, using security principles and practices in a documented networked environment, using all known system patches for security, and following good management practices to maintain a system information policy. Most applications will be commercial, i.e. the foundation will be commercial packages, but increasingly the application must be customized to fulfill a specific business process need. Customization may take many forms, and the coding language used by custom applications will affect the security of the resulting system. Highlights of these coding languages follow.

C and C++ are widely regarded as portable languages that allow applications to move across platforms. Compilation options and nonstandard terms may create debugging problems.

Common Gateway Interface (CGI), Practical Extraction and Report Language (PERL), JavaScript, Microsoft Macro Language, and similar scripting languages are very powerful, with cross-platform capabilities. Their power makes these languages good targets for hacking attacks, as they support both local and network capabilities.

Java is billed as cross-platform, but as with C and C++ great care must be used in writing actual code to ensure cross-platform capabilities. The Java language is somewhat unusual in having a security model (the sandbox), but the concept greatly limits the usefulness of some applications. Efforts to expand the sandbox are making Java more like ActiveX, providing more capability, though at greater risk, and with some user trust of the software provided through interfaces and signed code.

ActiveX is a Microsoft-unique language/capability for distributed custom applications. Though ActiveX is very powerful, the security model is fairly simple, based on signed code with authenticated signatures. Its flexibility is a concern to many security professionals.

There are other languages available, on various platforms, with other concerns. The four software applications considered in this section are generally assumed to be well-written code from developers lacking evil intent. The environment assumes that the vendor code functions as intended, without bugs.

7.1.1.2 Operating System Environment

This section of the framework is focused on the security services that applications could provide to protect data that the applications manage and manipulate on behalf of users. This data may be intended for private, narrowly shared, or widely shared consumption. Typically, an OS allows users to share hardware resources. The OS virtualizes and manages access to memory, disk drives, data ports, and other hardware resources. Its management separates users so that one user’s memory space cannot be read by another user’s process. The OS management also allows for portability, so that software code written on one machine may be ported to another machine with less difficulty than if all code directly called the hardware.

An OS provides several basic mechanisms to support information system and application security. The requirements for these mechanisms have been widely written about in the common operating environment (COE) requirements and in the Common Criteria (CC). A specific set of requirements for OSs is being captured in Common Criteria protection profile format through the Defense-wide Information Assurance Program (DIAP) to document requirements for protection of host computer OSs (clients and servers).

The OS environment should make it possible to securely identify and authenticate users of the system. Access controls and permission should be issued to all users of the operating system to ensure proper access to files and directories. The OS should also have an audit log, to provide security check points. An audit log can be used by system security administrators to backtrack system access if there is a security violation. The audit log itself must be well protected from unauthorized access and modification.

In selecting an OS, a risk assessment should be done to check vulnerabilities. This assessment can be especially necessary when an OS regularly receives patches. Failure to update patches regularly can leave an OS widely susceptible to hackers and other security breaches.

7.1.1.3 Standards and Protocols for Providing Security to System Applications

Efforts at standardizing security features and services have attempted, as a primary goal, to specify algorithms, formats, protocols, configurations, etc. If there is standardization, the common security services (Section 4.4, Important Security Technologies) can protect against the universe of threats (Chapter 4, Technical Security Countermeasures) with maximum interoperability (Section 4.6, Interoperability Framework).

From an environment standpoint, the Information Assurance Technical Framework (IATF) emphasizes the importance of using open standards and COTS solutions. Commercial implementers are more and more dedicated to generating and using open standards that allow multiple independent implementations to interoperate. The security community is demanding public disclosure of the details of security protocols and algorithms so that these standards may be tested to an appropriate level of assurance.

The term “standard” is used quite loosely in the IATF. It is meant to include any standard, or any technology or product initiative that could evolve into a standard. Standards can encompass national, international, Department of Defense (DoD), federal, allied, and commercial standards. This framework primarily addresses standards relating specifically to security but may also include other standards that affect interoperability or system infrastructure. Security is often simply an element of a broader standards activity.

Specific examples of standards and protocols of interest include the following (see also Section 4.4, Important Security Technologies).

- Application layer
  - Secure Hypertext Transfer Protocol (S-HTTP)
  - Object Management Group’s Common Object Request Broker Architecture (CORBA)
  - W3C XML Transfer Protocol
  - Secure File Transfer Protocol (S-FTP)
  - Secure Electronic Transactions (SET)
  - Message Security Protocol (MSP)
  - Secure/Multipurpose Internet Mail Extensions (S/MIME)
- Transport and network layer
  - Transport Layer Security (TLS)
  - Secure Sockets Layer (SSL ver 3.0)
  - Secure Shell (SSH)
  - Internet Protocol Layer Security (IPSec)

- Data link layer
  - Point-to-Point Protocol (PPP)
  - Serial Line Internet Protocol (SLIP)

- Security management infrastructure
  - Internet Engineering Task Force (IETF) Public Key Infrastructure (PKI)
  - IETF Simple Public Key Infrastructure (SPKI)
  - IETF Domain Name System Security (DNSSEC)

- Data labeling
  - National Institute of Standards and Technology (NIST) Federal Information Processing Standard (FIPS) 188 Standard Security Label
  - Institute of Electrical and Electronics Engineers (IEEE) 802.10g Secure Data Exchange (SDE) Security Label
  - IETF Internet Security Label
  - International Organization of Standardization (ISO) SC-32 Security Label
  - Military Standard (MIL STD) 2045-48501 (Common Security Label)
  - SDN.801 Reference Security Label
  - ISO MHS X.411 Security Label.

7.1.2 Consolidated Requirements

Security requirements for applications can be divided into two areas: functionality and assurance. The application security functionality requirements are simply a list of the security functions the application must supply if the information on the system is to be protected. Functionality requirements can usually be specified and tested objectively. The functional requirements for application layer software are broad—and range from local applications to all the many different approaches to communication and collaboration between users. The difficult question is where the requirements should be levied, in the OS or the application program. Common, widely used functions, such as file system access control, belong in the OS, specialized functions, are in applications. High-level functional requirements include the following:

- The application must be user-friendly, with well-documented user interfaces.
- The application must use correct and efficient backend processing.
- The application must support standards and implementation with standards-based API.
- The application must protect the privacy and integrity of user and system data.
- The application must authenticate the user to assign accountability.
- The application must generate a log of user activity for administrative monitoring purposes.

Management of configuration information should be centralized where possible and supported by secure remote management when necessary.

Assurance is a more subjective requirement. Assurance is a measure of confidence that the security features and architecture of an information system accurately mediate and enforce the security policy. Assurance requirements provide confidence that an application meets its security goals.

There are many different approaches to assurance. Process assurance requires the software developer to adhere to a specified software-engineering life cycle. Product evaluation assesses the design and realization of a product before approving it for use. Assurance provides increased confidence in the “goodness” of a product’s security features.

The National Information Assurance Partnership (NIAP), a U.S. Government initiative, intended to foster the availability of objective measures and test methods for evaluating the quality of information technology (IT) security products and accreditation of laboratories that can provide evaluation and validation services. In the United States, NIAP-accredited facilities contract with application developers to evaluate security products using methods and standards dictated in the Common Evaluation Methodology (CEM) for IT Security and the CC.

NIAP ensures that products meet the requirements and assurance levels proposed in security targets or a protection profile. Therefore, NIAP's duties are as follows:

- Evaluate application (using a standard, mutually agreed-upon process for reusable results—i.e., CC).
- Produce and monitor product evaluation test reports.
- Award NIAP-approved EAL certificate.
- Maintain lists of products evaluated.
- Standardized testing criteria and procedures.

7.1.3 Potential Attacks

The four classes of attacks are active, passive, insider, and distribution. These classes are a concern for security-enabled applications. Specific attacks, once identified will fall into one of these attack categories and can only be countered at a lower design level. Details of these attacks to application security are provided here.

7.1.3.1 Active Attacks

Protocol exchanges between clients and servers are common in application security. These protocols may have security as their immediate concern (authentication protocols) or they may provide application functionality, with the assumption that security is already in place. Many forms of spoofing and network connection hijacking have been observed; vulnerabilities have been identified in security protocols that were widely believed to be correct.

7.1.3.2 Passive Attacks

Passive attacks can vary greatly. Information collected may be clear-text or encrypted. Encrypted information may later be subjected to crypto analysis. Information passively captured may be used to support network replay attacks.

7.1.3.3 Insider Attacks

Attacks launched by trusted users inside an enclave are considered insider attacks. Insiders may be employees, contractors, service providers, or anyone else with legitimate access to a system. A cleared insider is a person who holds a clearance and has physical or administrative access to classified automated information system (AIS).

Protecting against and detecting malicious behavior by insiders is one of the most difficult IA challenges. Both technical and procedural countermeasures can reduce the risk, but to be effective technology and procedures must complement one another. Countermeasures to this form of attack include enhanced background checks, physical security, and limiting each individual's authorized privileges. The application and the security features it provides can also partly counter these threats with features such as audit, two-person administrative requirements, and covert access prevention and detection.

7.1.3.4 Distribution Attacks

Because the risk of malicious code in commercial application software is difficult to quantify, it is difficult to judge the value of countermeasures. For mass-produced office application software, which can be obtained from many sources, the risk of malicious software hidden in applications must be considered. For custom applications created for security-conscious organizations, the malicious software risk may be addressed in the design and development of the software. Defensive options include review and control of the source code and security requirements on the software development process.

7.1.3.5 Lower Level Attack Analysis

Poor protocol specifications may enable lower level attacks. Careful analysis of the specifications of protocols such as Transmission Control Protocol (TCP) and Server Message Block (SMB) can identify opportunities for attacks that compromise information or deny service. For instance, the draft SMB protocol specification includes its security limitations.

Beyond the protocol specification, specific implementations can enable attacks. For example, some implementations that use a simple predictable algorithm to generate initial sequence numbers are susceptible to a well-known spoofing attack.

Another common group of lower level attacks exploits the failure of application code to do memory bound checks or other error analysis on data provided by external sources. Buffer overflows and other tricks can then be used to cause malicious remote command execution.

7.1.4 Potential Countermeasures

Because information systems can be susceptible to attacks at many levels, countermeasures must span a similar range. Some apply to the entire system. Some are application specific. At the most basic level some must respond to implementation-specific attacks.

Countermeasures must continually be improved to counter more sophisticated attacks. The ultimate goal is for the countermeasure to become so sophisticated that the cost of mounting the attack exceeds its potential value if successful: The threat to an information system is reduced when the rational attacker discovers the reward does not warrant the effort of the attack.

Countermeasures are enabled through various security mechanisms, such as cryptography. Cryptographic mechanisms include public key certificates, key exchange (public key cryptography), data encryption (private key cryptography), digital signatures, and secure hashing. Chapter 8, Supporting Infrastructures, is devoted entirely to supplying keys and certificates for cryptographic mechanisms and the infrastructure for managing keys and certificates. The chapter deals with PKI/certificate management infrastructures (CMI), and security management infrastructures (SMI) and the capabilities, security considerations, and policy that pertain to them. Functionally, PKI, CMI, and SMI are intended to authenticate that a certificate is tied to a unique entity, secure distribution of certificates and private key material, wide distribution of public key material, and notification of compromised and revoked certificates or key material. Technical and policy measures that counter attacks and security concerns related to key management are detailed in Chapter 8.

Details of security services and the countermeasures they provide are described in the following sections.

7.1.4.1 Access Control

Access control is the process of granting access to information and information systems only to authorized users, programs, processes, or other systems. Access can be controlled by identification and allotted roles, roles alone, user name, group membership, or other information known to the system. A well-managed Windows or UNIX OS can provide basic access control that limits user access to specific resources and privileges.

Controlling access to system resources, such as one of its applications, protects the data associated with the resource. Those who intend to alter the resource's information or add a malicious process are foiled because they are denied access to that data by the OS. It is particularly important to control who may enable or disable (turn on or turn off) the security features that may be built into the application or change programs or the privileges of users.

Secure applications that process data must be aware of their role in managing access to that data. That includes knowing who is attempting access, mediating access according to processing rules, auditing user actions, and managing where (access to printers in particular locations) or how (encrypted channels like SSL) data is sent. Access control may be managed solely by the application, or it may use OS functionality for assistance, as when a database uses OS controls on files (user/group/world read/write privilege) by putting different classes of information into different files with different access privileges, with the users directly accessing none of the data.

7.1.4.2 Identification and Authentication

Identification and authentication (I&A) is the process of identifying and authenticating the identity of the user who is trying to access a system, thus providing accountability. When I&A is used with effective access control, the more uniquely the user can be identified and the more assuredly this identity can be authenticated, the more secure the system.

Identity can be assured by requiring the user to show something he or she has (e.g., an identification badge or a hardware token). That identity can be authenticated by requiring the user to provide something he or she alone knows, (e.g., a password or personal identification number [PIN]) and something uniquely his or hers (e.g., a fingerprint, retinal scan, or other biometric).

Electronic or digital signature can also authenticate users. A public key certificate—an electronic certificate signed by an issuer—that can provide a unique digital identity for the holder of the certificate. Validating the certificate chain is part of the authentication process. The certificate issuer authenticates the identity based on possession of the certificate issued.

7.1.4.3 Data Integrity

Data integrity means that data is maintained as intended and has not been exposed to accidental or malicious modification. Data integrity is separate from data encryption, although some encryption algorithms can be used to prove that integrity has been maintained.

An OS and an application can work together to protect data from modification. The OS can provide integrity on its files; they can be saved, opened, modified, and closed by applications with the assurance from the OS that the information on the files is changed only if an authorized application made the changes.

The application and the OS can provide additional integrity through use of a secure hash function: Each entry is mathematically hashed, producing a unique value for that entry. Verification of the hash guarantees integrity of the data. A digital signature applied to the hash value authenticates the hash value and who applied it. It is important to note that hashing is a one-way function; the hashing algorithm cannot be reversed to reconstruct the data from the hash value.

7.1.4.4 Data Confidentiality

Data confidentiality means that information is not disclosed to unauthorized entities or processes. Access control mechanisms support data confidentiality in information systems by controlling access to the system's resources.

Confidentiality is especially important when the application is not running. Without the OS or application controlling access, data in storage is especially vulnerable, as is data in transit, outside the direct influence of its generating application. Encryption is useful in both cases. Both applications and OSs can encrypt stored data and data in transit. Data confidentiality is directly related to the algorithm used to encrypt data and the protection of the key used for encryption.

7.1.4.5 Availability

Availability means that the adversary does not deny the access and processing of data to authorized users. Data that is inaccessible might as well not be there. Likewise, applications that fail to work are useless. The OS and applications should be designed to withstand failure in either the OS or an application. Most UNIX systems and Windows-based systems have error handling routines and fault isolation, making the OS more available if there is an application failure. Applications should be designed and tested to ensure that they do not fail, particularly under extreme conditions—the robustness of an application cannot prevent problems when the underlying OS or external network components (guards, firewalls, routers, cable) fail.

7.1.4.6 Nonrepudiation

Nonrepudiation means that the recipient is assured of the originator’s identity and the originator is provided with proof of delivery, so that neither can later deny having processed the data. Nonrepudiation counters man-in-the-middle and spoofing attacks.

One way to achieve nonrepudiation is with digital signatures and auditing. Before transmitting, the originator signs the data with an algorithm that incorporates parameters unique to the originator. Verifying this signature verifies the originator’s identity. Auditing makes a complete record that can serve as evidence and protects the record’s integrity. For proof of delivery, the originator when sending data requests a signed receipt. The recipient signs it with an algorithm that incorporates parameters unique to the recipient. Verifying this signature verifies the recipient who received the data.

Since nonrepudiation often depends on an identity contained in a public key certificate, which can become invalid, it is important that a trusted third party be able to validate the certificate. It must be possible to prove the validity of the certificate at the time of the original communication, and the authentication must be recorded in the audit trail.

7.1.4.7 Auditing

Both the application and the OS can audit certain actions taken by users and software acting on the OS. An application might track when a user enters data into a database and information related to the data or its position in the database. An OS might track which users initiate a process or attempt to access certain files. Auditing is primarily an after-the-fact activity that supports information forensics activities and intrusion detection. To detect intrusions into a computer or network, tools are available to observe security logs or audit data. These tools can be integrated into either the OS or the application, or can be separate software added to a system. See Chapter 6, Defend the Enclave Boundary/External Connections, and Section 7.2 for an in-depth discussion of intrusion detection.

Auditing is a protective measure only in the sense that knowledge that there is auditing may deter some threats to information systems. Auditing is much more useful in detecting questionable activity and reacting to such activities.

Applications developers should make explicit use of OS audit capabilities and plan for the use of the audit data by system administrators or other security professionals. One of the overarching technology gaps today is the lack of useful audit tools.

7.1.5 Technology Assessment

Three technology areas—cryptographic security services, applications, software download, software update, and biometrics—will be considered separately.

7.1.5.1 Cryptographic Security Services

If applications are to use cryptographic security services, first some type of cryptographic algorithm must be available. This framework will assess, not specific algorithms, but the medium, the token, on which an algorithm is presented to the application for use. The algorithm on the token is presented through a CAPI.

7.1.5.1.1 Cryptographic Tokens

Stand-alone cryptographic devices met the security needs of the past. Confidentiality was the security service of choice; it was implemented with link encryption, with one device servicing many users.

The need for security services beyond confidentiality has arisen with the growth of network technology. One such needed service is I&A—the need to specifically name users and have assurance that the persons associated with those names are who they claim to be. As cryptographic technology has progressed in both size and cost, it can now provide personal security services. Each user can have a cryptographic device (security token) that is uniquely his or her own. Tokens can also provide data integrity and nonrepudiation services through hashing and digital signature algorithms.

Using a personal security token that implements public key cryptography enables each user to have a unique private key that can be used as the basis for the security services of nonrepudiation and I&A. One way to accomplish this is to use the keys to create digital signatures on messages. The recipient of such a signed message can verify the digital signature before accepting that the message is truly from the user who claimed to send it. Tokens can come in different forms—from Personal Computer Memory Card International Association (PCMCIA) cards to smart cards and even software. Each offers advantages and disadvantages.

PMCIA Tokens. A PCMCIA security token can offer a full suite of portable security services. Board real estate allows room for sizable RAM and electrically erasable programmable read only memory (EEPROM) or Flash EEPROM, providing ample memory for complex or multifunction firmware and certificate storage. Since it is a hardware token, the PMCIA card can also protect secret values reasonably well and still leave room for additional physical tamper-protection mechanisms. On the downside, PCMCIA cards require PCMCIA card readers, which, although they are prevalent in laptop computers, are not common in desktop computers. The added expense of a card reader for every desktop workstation is definitely a disadvantage of the PCMCIA token.

Smart Cards. The smart card offers the same portability as the PCMCIA token at less cost. These cards still require special readers but the readers are much less complex than PCMCIA readers and therefore less expensive. Some manufacturers are incorporating smart card readers into their computer keyboards.

One significant concern with smart cards is data throughput. The defined interface is just too slow to support confidentiality services for any but the least demanding applications. Confidentiality would normally be relegated to software running on the workstation, which can reduce the assurance of this service; I&A, nonrepudiation, and data integrity would remain in the hardware on the smart card.

Software Tokens. Software tokens are the cheapest but also the least assured solution. Their implementation in software allows for quick distribution, ease of updating, and responsiveness to the needs of most users without the need for special hardware. When the security solution calls for minimal assurance and when cost is a major consideration, software tokens could be the answer.

There is a price to be paid with software, though: Software tokens will execute on untrusted workstations running untrusted OSs that make them ultimately vulnerable to bypass, modification, or even replacement. Systems that process highly sensitive information should not rely solely on software tokens security.

7.1.5.1.2 Cryptographic Application Programming Interfaces

As application developers become aware of the need for cryptographic protection, they add “hooks” to access cryptographic functions developed by others. These hooks at the lowest level (sometimes crossing into the OS and almost always within what would be called middleware) are the CAPIs. As CAPIs become more sophisticated, their value increases. Applications that use a standard CAPI can access multiple cryptographic implementations through a single interface. This helps to minimize life-cycle implementation efforts, and cryptographic modules built to a standard CAPI can be accessed by a greater number of applications, increasing reusability.

The numerous current efforts to create CAPI standards range from the very basic, like that found in Generic Security Services (GSS)-API, to those more directly controlling the cryptographic token, like Public Key Cryptographic Standards (PKCS) #11, and increasingly applications and cryptographic modules are being written to use certain CAPIs. While a single standard usable by all applications would be ideal, multiple CAPIs are required to support the broadest range of applications and cryptographic modules.

CAPIs are intended to provide these features—

- Interface between cryptography and applications
  - Facilitate the development of new security-enabled applications
  - Minimize cryptography processing by the application
- Application independence—support a broad range of application types: store and forward and connectionless.
- Module independence—support the entire range of hardware and software tokens.
- Algorithm independence—support a broad range of current and future algorithms.
- Functional completeness
  - Provide comprehensive security services
  - Facilitate cryptography export policy.

High Level: GSS-API. The GSS-API and the extensions for independent data unit protection (IDUP) support applications that do not interface with cryptographic services. These Microsoft security service providers (SSAPI) provide a high-level interface to authentication, integrity, confidentiality, and nonrepudiation (IDUP-only) services. The application merely indicates the required security services and optionally the quality of protection (QOP) for the per-message services.

GSS-API was designed to protect session-style communications like File Transfer Protocol (FTP) between entities. IDUP-GSS-API does not assume real-time communications between sender and recipient. It protects each data unit, whether file or message, independently of all others. IDUP-GSS-API is therefore suitable for protecting data in store-and-forward applications. The specifications for it were developed within the Common Authentication Technology (CAT) group within the Internet Engineering Task Force.

Mid-Level: CDSA, MS SSAPI. The Common Security Services Manager API (CSSM-API) is the heart of the common data security architecture (CDSA). CSSM-API offers a robust set of security services, among them cryptography, certificate management, trust policy, data storage, and optional key recovery. CSSM-API can support auditing services and provide integrity services via the Embedded Integrity Services Library (EISL).

CSSM-API, developed at Intel Architecture Labs, is approved as a standard within the Security Program Group (SPG) of the Open Group (the result of the X/OPEN and the Open Software Foundation merger). While such CSSM services as certificating management, trusting policy, and data storage fit logically at the middle level, the actual CAPI calls (their cryptographic service provider interface [SPI]) are more low level, like Cryptoki. For instance, CSSM-SPI supports user authentication and administrative control of tokens.

SSAPI is modeled after the GSS-API, though with more of a Windows style. It provides mutual authentication, message privacy, and message authentication because it is connection oriented, it is used for protocols defined by Microsoft as "SChannel"—SSL and WinPCT. It also supports NTLM, DPA, and Kerberos.

Low-Level: Cryptoki (PKCS-11), Cryptographic API (CryptoAPI), Cryptographic Interface (CI) Library. PKCS #11–Cryptoki is an OS-independent abstract token interface that defines the arguments and results of various algorithms. Cryptoki also specifies certain objects and data structures that the token makes available to the application; it interfaces directly to cryptographic tokens and is thus the logical place for functions that allow user authentication (e.g., logon or PIN entry) and administrative control of the token. Cryptoki, developed by RSA Labs and a member of their family of PKCS, is appropriate for use by developers of cryptographic devices and libraries. PKCS #11 workshops sponsored annually by RSA Labs for all interested parties contribute to the continuing development of Cryptoki.

As a service suite provided by the Windows NT OS, CryptoAPI provides extensive facilities for both hardware and software cryptographic modules, called cryptographic service providers (CSP). CryptoAPI has not been subjected to any formal standards process, but the authors at Microsoft did consult with various government and corporate customers. Applications using CryptoAPI can take advantage of default features of the interface to reduce their cryptographic awareness requirements, or they can exert full control over algorithms, keys, and modes of operation. Tables 7.1-1 to 7.1-3 depict specific pros and cons for GSS-API, CDSA, and Cryptoki:

The FORTEZZA® CI Library was initially the interface between the FORTEZZA PCMCIA card and applications wishing to use the security features associated with the National Security Agency's (NSA) Multilevel Information Systems Security Initiative (MISSI) program. The CI Library is now being adapted for both smart card and software token implementations of FORTEZZA.

Table 7.1-1. Pros and Cons of GSS-API

Advantages	Disadvantages
Abstracts the lower level details of such services as cryptographic routines and key management.	GSS-API services must be implemented for each technology (e.g., Kerberos, Cryptoki).
Is platform independent—written with low-level programming languages (C/C++) as well as platform-independent languages (Java).	Complicated implementations require experienced developers to reduce the learning curve.
Is constantly being updated to match changes in technology.	
Is flexible enough to allow for addition of current technologies, such as PKCS #11.	

Table 7.1-2. Pros and Cons of CDSA

Advantages	Disadvantages
CSSM provides an "all in one" interface to security services (privacy, authentication, integrity, and non-repudiation).	It requires lower-level work to be done by plug-in modules. There must be trust that these modules are implemented correctly.
It is expandable to future CAPI implementations via the elective module manager.	Support is provided, not by Intel, but by the Open Group—mostly users, not developers.
It is designed specifically to deal with network operability and security solutions.	It is complicated to implement solutions.
APIs allow for hardware tokens, software modules, and hybrids of the two.	
It calls mimic security calls from previous non-standard implementations, allowing easier transition from nonstandard APIs.	
It has already implemented a PKCS #11 layer.	

Table 7.1-3. Pros and Cons of Cryptoki (PKCS #11)

Advantages	Disadvantages
Allows use of broad range of token-based devices (hardware and software).	As a stand-alone application, allows only for peer-to-peer security service.
Is compatible with middle and high-level APIs, such as CDSA and GSS-API.	Requires a user to log in for communication between software and token device—there is no built-in key infrastructure.
Interface is intuitive object-oriented (OO): public and private objects with attributes and methods, allowing easy modeling within such popular low-level OO programming languages as C++ and Java.	Requires token manufacturers to conform to the PKCS #11 standard.
Is compatible with different key types (RSA, DSA, Diffie-Hellman, RC2, RC4, DES).	

7.1.5.2 Applications

Applications are generally useful for exchanging information among many people within a specific system, or between information systems. The applications discussed here are "mission" applications; the basic functionality has been adapted to meet a particular mission need. Examples include databases, collaborative computing applications, and electronic commerce systems. Because information is being transmitted, the need for standard, interoperable, and secure applications is critical. While many applications are mature, most do not support the broad range of security services.

7.1.5.2.1 Mission-Specific Applications

Mission-specific applications can be as simple as a database making its data available through a fronting web server. They can be as complex as a complete travel service that checks and books airline, hotel, and rental car reservations through a Web browser; passes the information to the user via e-mail; and keeps the whole system secure with file encryption. These systems typically rely on existing COTS products, such as Web servers and clients and database management systems. As security is only one of many factors in the selection of such products, many desirable security features may not be present. In addition, legacy systems with very little security must often be included as part of the solution.

For mission-specific applications, which must enforce a definition of security unique to the application and the circumstances of its use, the security challenge is to combine many less-than-ideal generic component-level security services into a cohesive, meaningful application-level definition. This is a significant information system security engineering task.

Mission applications are often custom built in several distinct tiers. The three-tier model typically has a presentation layer, a business process layer, and a database layer. A conventional client/server system uses a two-tier approach. A system can have multiple separate application layers creating multiple tiers (see Figure 7.1-1). Collectively these systems are referred to as "n"-tier systems. Different systems will place different numbers of layers between the user and the data, and some may simultaneously support multiple paths to access data. There are many ways in which a mission application can be secured using readily available technology. Some of these enable the construction of new security-enabled systems. Others allow security to be retrofitted to existing systems or components. All are extensions of the security provided by the various system components.

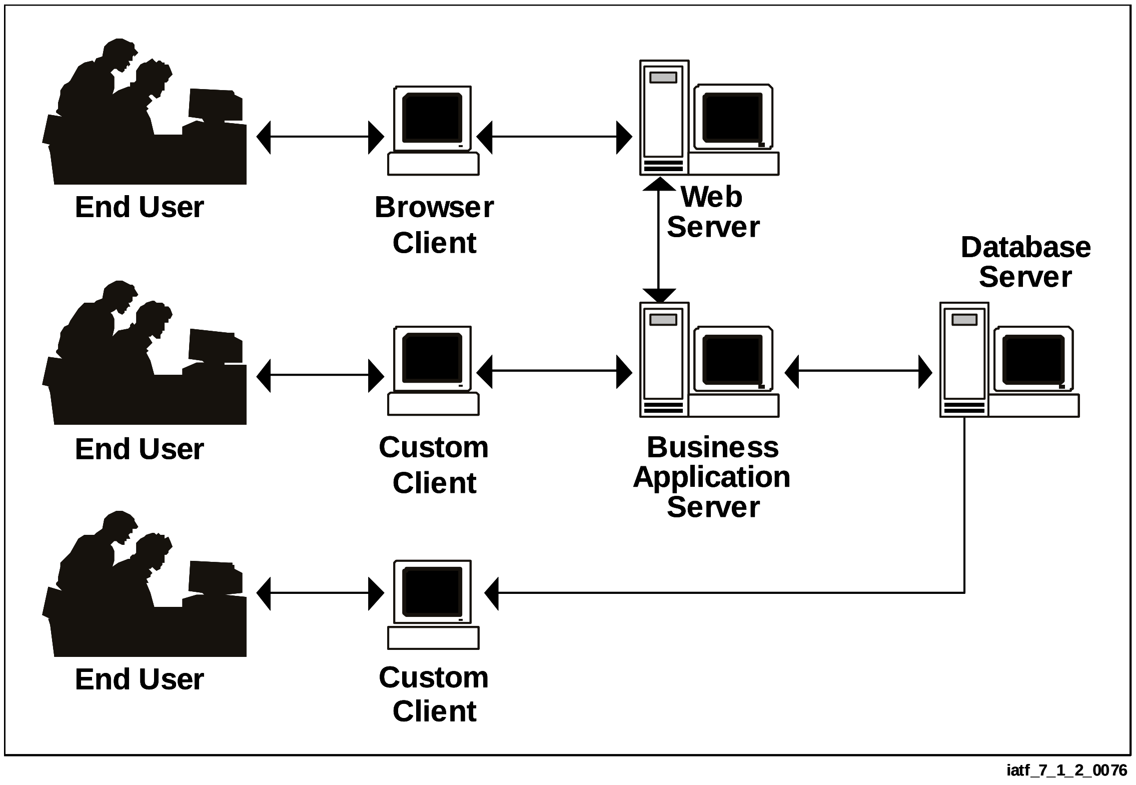

Figure 7.1-1. Custom N-Tier Applications

7.1.5.2.2 File Protection

File encryptors protect information in the computer if there is unauthorized physical access by encrypting the stored information. There are two basic types of file encryptors: one in which the user selects specific files to encrypt and one that automatically encrypts all information that is not currently being processed. The former can be used to securely transfer files as attachments or to protect critical information stored on floppy disk, CD, or a user's system. The latter are often referred to as media encryptors.

Media encryptors encrypt the entire contents of the drive except for some system files that must be left unencrypted so that the computer can boot. The integrity of most of these system files can be protected by a cryptographic checksum; this will not prevent a tamper attack, but it will alert the user that the data has been altered. Some system files, however, contain data that changes when the computer is booted. These files cannot be protected at all. The mechanisms implemented by media encryptors provide—

- Encryption of system files.
- Integrity of the contents of the data storage media.
- Confidentiality of the contents of the data storage media.

- Integrity of the workstation, verifying the basic input/output system (BIOS) and ensuring that configuration and program files are not modified.
- Recovery of data if the original user can no longer access the media.
- Key management support: key generation, distribution, deletion, destruction, and revocation.

File encryptors typically implement a GUI that allows users to choose files to be encrypted or decrypted. This protects individual files but not all the files on the drive. The mechanisms implemented by file encryptors provide:

- Encryption of selected files.
- Integrity of the contents of the protected file.
- Confidentiality of the contents of the protected file.
- Authentication of a file's source.
- Exchange of encrypted files between computers.
- Recovery of data if the original user can no longer access the file.
- Key management support: i.e., Key generation, distribution, deletion, destruction, and revocation.

Many applications generate temporary files that may contain user data. These files are normally erased when the application is closed, but when the application does not close in an orderly fashion, these temporary files may remain. Some OSs do not actually erase data when files are deleted; instead, they alter the name of the file in the file allocation table. The user's data remains on the hard drive until the space is reallocated to another file and overwritten. Thus, after system shutdown, unencrypted and potentially classified user data can remain on the hard drive, because of either failure to erase temporary files or the design of the OS's erasing function.

The Range of possible architectures for the KMI/PKI needed to support file protection is wide. Possibilities range from a user having complete control over key generation and distribution to a hierarchical architecture involving a complex certificate authority (CA). KMI/PKI is discussed in detail in Chapter 8, Supporting Infrastructures.

7.1.5.3 Software Download

Planning for the secure update or download of software must begin early in development and continue throughout deployment. Three types of software download will be considered: firmware updates, software updates, and new software distribution. In all cases, the most critical aspects of downloads are the integrity of the downloaded software and authentication of the origin of the software. Sometimes confidentiality of the download may be required. Validity periods, usage limitations, effects of the download on system data, and auditing of the download installation may also be important.

7.1.5.3.1 Firmware

The key to managing firmware updates, exemplified by a recent update of modem software to support a new 56k standard, is planning for the hardware's ability to support that update: That hardware's ability must verify the integrity and authenticity of the firmware originator and the originator's associated firmware.. Because firmware is being updated, it can generally (but not always) be assumed that the firmware will be processed by the hardware during installation. In general, hardware processing is preferred over software processing because hardware is faster and has greater resistance to tampering.

Planning for a firmware update must begin with initial product development. Steps that must be taken during initial product development include the following:

- Decide what security services firmware update requires.
- Choose mechanisms to implement chosen security services.
- Confidentiality or integrity services may use a cryptographic mechanism.
- Cryptographic mechanisms include symmetric and asymmetric encryption.
- Determine whether symmetric or asymmetric cryptography will be used.
- If asymmetric encryption, generate a public/private key.
- Make the public key information readily available.
- If symmetric encryption, generate and store symmetric key material and determine secure distribution process to the user base see section 8.1.
- Field the initial product.

Updating the fielded product requires the firmware developer to take the following steps:

- Generate the code that updates the previously installed firmware.
- Cryptographically hash the updating software.
- Sign the hash with the appropriate keying material.
- Encrypt the package (software, hash, and signature).
- Distribute the package.

The deployed system user should then use the appropriate keying material to verify the signature and integrity of the firmware update. Then install the update package. Update status to include failures of the signature or integrity of the firmware update should be reported through a host user interface.

Integrity—Package integrity is provided by cryptographically hashing its contents. See section 7.1.5.4, Software Update for more detail.

Authenticated Origin—Signing the hash provides proof of origin; the private aspect of the public/private key pair must be appropriately protected. See section 7.1.5.4, Software Update for more detail.

Confidentiality—Encrypting provides confidentiality to the firmware updates. It is more efficient to use symmetric cryptography to support confidentiality mechanism and asymmetric cryptography to support key distribution for the symmetric cryptography. The user's public/ private key pair creates a single-use private symmetric key for each download. See section 7.1.5.4, Software Update for more detail.

Other Security Services—Other security services can be provided by hardcoding information in the initial package or including information for processing in the package. For example, limiting use of an object to a specific time period could be handled by validity dates on the signature, coding in the object broker to allow a fixed period of use on each download. In all cases it is important to keep the security objective in mind and to manage a chain of trust till that objective is achieved.

7.1.5.4 Software Update

Developers distribute modifications to software that already resides on a system. These modifications include service updates to software packages such as Windows or Microsoft Office and distribution of active content code (e.g., Java, ActiveX, objects in Distributed Component Object Model [DCOM] or CORBA, macros, etc.). During the download some known trusted piece is already in place to verify the security.

Software updates and active code distribution are managed much like firmware updates, except that software updates may not be able to rely on hardware storage of key material, so the level of assurance is likely lower than with firmware updates. For most active content, there is a virtual machine (e.g., Java Sandbox or macro interpreter) limiting or at least managing the operation of the active code.

7.1.5.4.1 New Software Distribution

New software is best distributed on media that are hard to modify like CD-ROMS, in tamper-resistant packaging with unique vendor identification, like holographic labels, which are widely used by commercial vendors to prevent fraud. Some software distributions include side programs to verify authenticity of the package, or are self-checking. However, since anyone can write code that appears to verify or self-check other code, these mechanisms are not particularly useful.

7.1.5.5 Biometrics

Biometrics is an authentication mechanism to support access control. A truly automated biometrics system should be able to discern a user at any terminal. The associated authorization service determines the correct access and monitors to ensure that only the authorized user accesses the information or information system. Access controls are policies or procedures establish criteria for system access. Identification service determines the identity of a user and authentication service verifies that identity. Authentication mechanisms fall into one of three types:

- **Authentication by Knowledge (Type 1)—**Something a person knows: passwords, codes, or PINs.
- **Authentication by Ownership (Type 2)**—Something a person owns or possesses: tokens, magnetic stripe cards, PCMCIA cards and smart cards.
- **Authentication by Characteristics (Type 3)—**Something that is a physical aspect of the person, including unique personal biometric characteristics such as fingerprint, retina, or facial.

The rest of this section will discuss Type 3, authentication by characteristics, also known as biometrics authentication. Biometric technologies include both the automatic collection and comparison of characteristics stored in an electronic medium and later used to confirm the identity of an individual. A typical authentication process consists of the following basic steps:

- **Enrollment or Capture Phase—**The actual biometric sample is taken from the user and stored in a database.
- **Feature Extraction Phase—**The appropriate measurements of the biometric sample are taken from the live scan of the user.
- **Comparison Phase—**The features extracted from the live scan are compared with the template stored in the database.
- **Decision/Evaluation Phase—**The processed data that has been compared is evaluated and given a score. Depending on the security threshold, access will either be granted or denied.

The methodology for integrating products into usable solutions requires directorates, customer requirements, a prioritization process, and viable solutions that culminate in a decision to accept, reject, or delete the request.

7.1.6 Cases

The potential for insider attacks alone makes it paramount for security mechanisms to be implemented for all applications and on all workstations. How strong these security mechanisms need to be depends on the damage a successful attack could cause. Cases can be defined based on the sensitivity (security classification) of a workstation user, the associated threat, and the enclave configuration. High-sensitivity workstations are assumed to employ complementary confidentiality, integrity, and availability mechanisms, e.g., strong authentication and encrypting and signing files and e-mail. As sensitivity-classification differences between workstations and individuals in an enclave increase, the need for the countermeasures increases. As the size of the enclave increases, the need for coordinating and managing its security similarly increases.

7.1.6.1 Cases Within the Enclave

The following cases represent different environments where security mechanisms are needed on workstation applications to protect information within the enclave boundary:

- Individual user with unclassified information that is personally sensitive within an unclassified enclave.
- Individual user with classified/restricted information within an enclave of equal sensitivity level.
- Subnet of users with unclassified information that is limited to these users within an unclassified enclave.
- Subnet of users with classified/restricted information within an enclave of equal sensitivity level.

7.1.6.2 Cases Transiting the Enclave Boundary

Although cases involving information transiting enclave boundaries are handled elsewhere in this framework, applications can further protect this information. The following cases represent environments where the application can provide this additional layer of protection:

- Individual user with U/SBU personally sensitive information communicating with an unclassified network, e.g., the Internet.
- Individual user with classified/restricted information connecting to a network of equal sensitivity level.
- Remote user connecting through a public network to an unclassified local area network (LAN) (remote access).
- Remote classified user connecting through a lower level network to a classified network (several subcases by deltas in levels) (remote access).
- Unclassified/sensitive/restricted but lower-value-information LAN connecting to a large, open, unclassified network, e.g., the Internet (many adversaries of varying capabilities).
- Unclassified or classified (valuable information) LAN connecting to a network of the same classification (less open).

- Classified LAN or LAN containing valuable information communicating through a lower level network to another network of equal classification (System High Interconnects).
- Classified LAN or LAN containing highly valuable information communicating with a lower classification network (High-to-Low, multilevel security [MLS]) (multiple subcases exist for varying deltas between information on the LAN versus the wide area network [WAN]).
- Classified LAN or LAN containing valuable information connecting to same classification/value/organizational WAN that has limited connections to lower classification/value/external network, e.g., secret LAN connected to a secret WAN that is also also connected to an unclassified WAN.
- Sensitive, restricted, or compartmented information LAN or subnet connecting to a corporate net or intranet.

The first four cases describe a single workstation connecting to a similar-security-level component, employing a potentially lower sensitivity transmission medium. Cases 5 through 7 are interconnected networks of essentially the same sensitivity level, employing unprotected (lower sensitivity level) transmission media. Cases 8, 9, and perhaps 10 involve high-to-low connections that may jeopardize interconnected high-level systems that are not aware of the low connection. Case 10 may involve a range of differences in information value of the subnet versus the network.

7.1.7 Framework Guidance

This framework characterizes the security features and assurances needed to protect information in today's richly interconnected environments. Applications process and circulate information, providing affordable security-enabled applications is therefore paramount to providing information assurance for the system. If implementing security-enabled applications involves significant financial investment, organizations and users will be reluctant to implement them. Developers must strive to create security-enabled applications that meet user needs without adding extras that drive costs to prohibitively high levels.

This section will not provide guidance for each case presented in Section 7.1.6, but will offer provide guidance that can apply in all cases. Specific requirements for each case and application type will be provided in the form of protection profiles that support the DoD Defense-in-Depth strategy.

7.1.7.1 User Interface

A security mechanism that is cumbersome to use will not be used. The importance of an intuitive and burden-free user interface for day-to-day operations cannot be overemphasized. The user interface also affects key management, both procedural and electronic, at least during start-up and it is important that it does not cause undue burden. If it does, encryption and digital signatures will not be widely accepted or used within the organization. The interface should keep the user apprised of security-related events and information, such as—

- Outgoing information that has been encrypted or digitally signed.
- Incoming information that is encrypted or digitally signed.
- The identity of the person who encrypted or digitally signed the incoming information.

7.1.7.2 Security Mechanisms

Not every vendor implements security mechanisms in the same way. Providing configurable options increases the chance that products from different vendors will operate together. These mechanism options can include the algorithms and associated key lengths supported by the application and the protocols used to transfer information between users, e.g., S/MIME or MSP for messaging. There must be a trade-off between the need for the secure application to support a number of options and the need for the application to be inexpensive and easy to use. Generic applications should be able to determine the mechanisms that are common when two or more applications attempt to interoperate.

There are two ways to add security mechanisms to applications: First, software plug-ins with security features can be added to existing nonsecure applications, or alternatively, security mechanisms can be directly integrated into the application during product development. Although there are advantages to both methods, the second is preferable. Security should be an integral part of an application, not an afterthought. The following is a list of constraints that security-enabled applications should meet:

- Applications with similar functions should interoperate, e.g., secure e-mail packages can communicate with different secure e-mail packages.

- The user has the choice to enable security mechanisms selectively for each message or file being sent.

- The user should be able to apply to information encryption only, digital signatures only, or both encryption and digital signatures.

The encryption and digital signature mechanisms (e.g., algorithms, key lengths, or random number generators) should be of sufficient strength and responsive to the current legal policies for the environment in which they will be used.

7.1.7.3 Certificate Revocation and Validation

A policy is needed for certificate revocation. The issues surrounding such a policy include what determines when a key should be revoked; who can request a revocation; what actions need to be taken once it is discovered that a received certificate has been compromised; where the list of revoked certificates is maintained; and how the list is disseminated. Electronic mechanisms must be in place to enforce the revocation policy. The security administrator should be able to configure the revocation enforcement mechanisms as needed to implement the site's policy.

Revocation is necessary when a certificate becomes invalid before its normal expiration date. Some reasons that a certificate becomes invalid are—

- User name change (e.g., marriage).
- User status change (e.g., termination of employment).
- Compromise or suspected compromise of the private key (e.g., loss of the token or fraudulent use).

If a private key becomes compromised, an information systems security officer or someone else responsible for the organization's computer security should be notified as soon as possible.

Revocation is the process of removing a certificate from operational status. The end user or responsible party can request revocation, as can any authorized personnel. The most common revocation method is through publication of a Certificate Revocation List (CRL). When a certificate number appears on a CRL, other users know that it is not to be relied on.

It is important for the CRL to be maintained in a location that is easily accessible to all users; the policy must establish the identity of the trusted central server and the circumstances under which the users must check with that server. For example, a CA is a component of a PKI that is responsible for maintaining and publishing CRLs. The CA prepares each new CRL using facilities on the CA server and posts the CRL on a directory server either in its complete form or incrementally. Incremental versions identify changes from the previous incremental release.

Another technique to check the validity of a certificate is dynamic-real time validation. A protocol that supports this is the on-line certificate status protocol (OCSP). For each validation, the relying party requests the status of the certificate from a revocation service, which maintains an unpublished list of revoked certificates.

As another measure to revoke a certificate, the certificate being revoked should be removed from the certificate repository.

7.1.7.4 Password Practices

A security policy must include good password usage practices for the site. FIPS Publication 112-1, "Passwords Usage," provides information on good password practices, among them minimum password length of ten alphanumeric characters, maximum period of password usage, and random words (nondictionary). Electronic mechanisms should be in place to enforce good password practices, particularly when the passwords protect private key information. The security administrator should be able to configure the password enforcement mechanisms so as to implement the password policy.

7.1.7.5 Technology Gaps

Though the tools, mechanisms, and services necessary for building secure applications are generally available, there are serious gaps. The gaps result mainly from the difference between known capabilities and needs and the solutions that are available. Finding a full vertical solution is quite difficult; it would include tokens, certificate infrastructure, and applications that all understand each other, use the same type of certificate with the same fields, and as needed, use the same standards for interoperability.

This ideal solution is nearly impossible to find today. The market is fragmented at virtually every horizontal level. Tool vendors use different algorithms, service vendors use different protocols, standards are not completely defined for interoperability, the certificate infrastructure uses different certificate extensions (sometimes with different meaning or intent), directory services and query modes vary, and the applications use different standards or different protocols. E-mail is an excellent example: the Defense Message Service uses MSP mail formats, the commercial world uses S/MIME and OpenPGP. Some applications use X.509 version3 certificates, others still use version1.

This gap in vertical solutions is expected to be filled as products from larger vendors (Sun, Microsoft, Lotus, and IBM) begin to appear, but in the meantime, vertical solutions are often proprietary and thus of limited interest to the Government. As the gap in basic solutions narrows, there will be more concern with the capability and security provided by the products, with some implementations simply being more robust than others. Security for new technologies (smart cards, PCMCIA cards, dynamic hypertext markup language [HTML], Virtual Reality Modeling Language [VRML], and others), though needed, may be lacking in the first generation of products. Testing, evaluation, and use will eventually disclose the real security gaps are in applications, and what can best be done about them.

References

1. Honorable Emmett Paige, Jr. Selection of Migration Systems. Assistant Secretary of Defense Memorandum. November 1993.
2. Linn, J. Generic Security Service Application Program Interface. RFC 2743, Version 2, Update 1, January 2000.
3. Adams, C. Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP GSS API). RFC 2479. December 1998..
4. X/Open X/Open Preliminary Specification: Generic Cryptographic Service API. draft 8, 20 April 1996.
5. RSA Laboratories. PKCS #11 v2.11: Cryptographic Token Interface Standard. November 2001.
6. National Security Telecommunications and Information Systems Security Committee, National Information Systems Security Glossary. NSTISSI No. 4009. 5 June 1992.
7. National Computer Security Center. An Introduction to Certification and Accreditation. NCSC-TG-029, January 1994.
8. National Computer Security Center. A Guide to Understanding Security Modeling in Trusted Systems, NCSC-TG-010. October 1992.
9. Microsoft Corporation. Application Programmer's Guide: Microsoft CryptoAPI. Version 2.0, August 2001.
10. NSA Cross-Organization Team. Security Service API: Cryptographic API Recommendation. National Security Agency. 1996.
11. Schneier, Bruce, *Applied Cryptography*, 2nd edition. John Wiley & Sons, 1996.

This page intentionally left blank.

7.2 Detect and Respond Capabilities Within Host-Based Computing Environments

Fundamental goals of the Defense-in-Depth strategy are—

- Prevent cyber attacks from penetrating networks and compromising the confidentiality, integrity, and availability of enclave information.
- Detect and respond effectively to mitigate the effects of attacks that do penetrate and compromise the network. The host computing environment is the final line of defense for the Defense-in-Depth strategy. The fact that these workstations and servers can be vulnerable to attacks through poor security postures, misconfiguration, software flaws, or end-user misuse must be factored into the protection approach.

While detect-and-respond technologies offer perimeter and access controls, authorized internal and remote users within an enclave can attempt probing, misuse, and malicious activities, particularly when they have been authenticated by a host computer either as an authorized user or by impersonating an authorized user.

Detect-and-respond capabilities are complex structures that run the gamut of intrusion and attack detection, characterization, and response. The detect aspects of detect-and-respond are actually measurement services. Intrusion detection, network scanning, host scanning, and the like are measurement functions that, continuously or periodically determine the effectiveness of the protection systems deployed. Detect capabilities do not protect, but the respond capabilities can change protection mechanisms (e.g., instituting automatic disabling of a user's account after too many failed login attempts) or deploy new protections (e.g., stronger authentication systems).

The local computing environment is the logical location for host-based sensors within an enclave. This section addresses host-based sensors, including those that operate in near real time and those that operate off-line. Specific host-based sensor technologies addressed in the framework are shown in Figure 7.2-1. Sections 6.4, Network Monitoring Within Enclave Boundaries and External Connections, and 6.5, Network Scanners Within Enclave Boundaries, provide similar guidance on network sensor

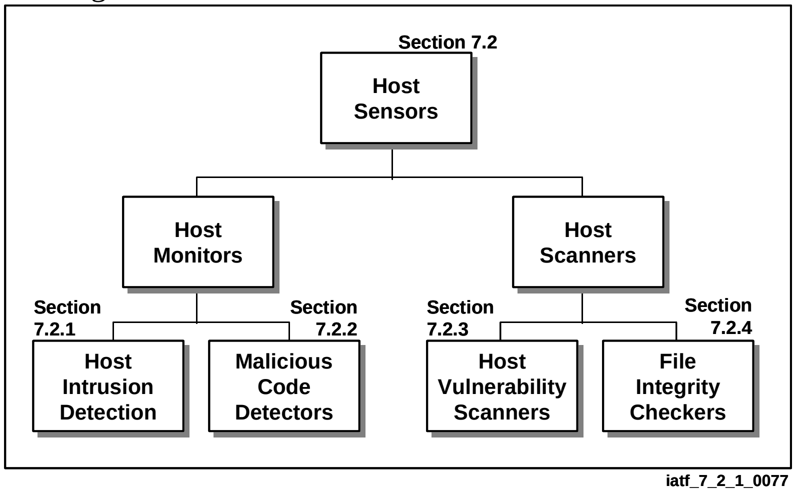

Figure 7.2-1. Breakdown of Host Sensor Technologies

technologies. There are common elements in the respective sections of the two chapters. Rather than cross-referencing the sections, each is structured as stand-alone for the convenience of the reader.

A number of functions (e.g., intrusion characterization and response formulation) are typically performed by analysts using the information provided by locally deployed sensors. Local environments may implement as much or as little above the sensors as they feel prudent, obtaining services and support from the system infrastructure as necessary. Section 8.2, Detect and Respond as a Supporting Element, discusses in-depth detect-and-respond processes in the context of information assurance (IA) infrastructure capability. It also offers guidance on technologies for processes beyond the sensors, though recognizing that they can be implemented at any level (including local) in an enterprise hierarchy.

Host-based sensors covered in this section include host monitors (intrusion detection and malicious code detector technologies) and host scanners (host vulnerability scanners and technologies for software integrity checking). The section reviews each relevant technology, general considerations for use, rationale for selecting features, and deployment considerations, and gives a perspective on how these technologies are typically bundled into products. The section concludes with sources of additional information and a list of references used in developing this guidance.

7.2.1 Host Monitors—Intrusion Detection

Today, most operating systems and applications generate an audit trail. Originally, it was intended that a security administrator would review the audit logs for suspicious events, but though this is current practice, the personnel typically available to review such logs are limited. Many enterprises do not use audit logs (or the tools to facilitate their analysis) for two major reasons. The tools themselves depend heavily on the user's ability to understand the types of attacks and vulnerabilities, and as the number of users, operating systems, applications, and databases grows, so do audit trail file sizes, which often consume too much storage space, possibly resulting in denial-of-service problems. Often, information technology (IT) operations staff are forced to delete or disable audit trails in order to avoid costly disruptions to their networks and information processing systems.

Technology Overview

The goal of a host intrusion detection system (IDS) is to identify, in near real time, unauthorized use, misuse, and abuse of computer systems by internal network users. As discussed in Section 6.4, Network Monitoring Within Enclave Boundaries and External Connections, similar structures and technologies are also available for performing comparable functions using network-based information.

Host-based intrusion detection sensors collect information in the form of the audit trail reflecting on a particular system. Information includes system logs, other logs generated by operating system (OS) processes, and contents of system objects not reflected in the standard OS audit and logging mechanisms. Systems can monitor information access in terms of who accessed what, map problem activities to a certain user identity (ID), and track behavior associated with misuse.

Host IDSs are based on the principle that an attack on a computer system will be noticeably different from normal system activity. An intruder, possibly masquerading as a legitimate user, is very likely to exhibit a pattern of behavior different from that of a legitimate user. The job of the IDS is to detect those abnormal patterns by analyzing the numerous sources of information provided by the system. Two major detection techniques are statistical analysis and rule-based expert system analysis.

- Statistical analysis attempts to define normal (expected) behavior. A popular way to monitor statistical measures is to keep profiles of legitimate user activities, such as login times, central processing unit (CPU) usage, favorite editor and compiler, disk usage, number of printed pages per session, session length, and error rate. The IDS uses the profiles to compare current and past user activity.
- Expert system analysis detects possible attacks on a computer system by searching for breaches of policy. It typically uses a rule-based system to analyze the audit trail records, trying to discover attacks based on the information contained in the rule base. The expert system can pose sophisticated queries to the rule base to answer conditional questions based on sets of events. These systems' main problem is determining exactly what the rules should be and what kinds of attacks can be detected by this method.

Detection Approaches

Anomaly and misuse detection attempts to separate benign from intentional unauthorized use of a system, applying special technologies to detect changes in the patterns of use or behavior of the system.

- Anomaly detection techniques assume that all intrusive activities deviate from the norm. These tools typically establish a normal activity profile, a statistical model that contains metrics derived from system operation, and then maintain a current activity profile of a system. Observed metrics that have a significant statistical deviation from the model are flagged as intrusive. When the two profiles vary by statistically significant amounts, an intrusion attempt is assumed.
- Misuse detection systems attempt to identify misuse of computing resources by authorized users. They look for exploitation of known weak points in the system that can be described by a specific pattern or sequence of events or data (the "signature" of the intrusion). For example, the user may be visiting unauthorized Internet sites, navigating around a system to areas explicitly identified as off limits, or using an application for activity unrelated to work. Misuse detection typically relies on an administrator's using configuration files to define activity that is considered misuse. The information in the configuration files can then be compared with an activity on the system; misuse is assumed when there is a match.

IDS Tuning Options

Typically, a host-based IDS provides capabilities for tuning its operation to a particular host and enterprise environment. Depending on the implementation, it is often possible to predetermine the types and specific attacks to be monitored, what the response will be for each detected intrusion (e.g., generate an alarm or record, or take a mitigating action), and characterize the class (e.g., the importance or severity) of each alarm generated. The IDS can be driven both by anticipated authorized activity on the host and the general information system usage characteristics across the enterprise. In this way, it is possible to focus the host IDS on specific events of interest, depending on what threats have been identified as relevant to the particular host environment and the response the IDS will have when events are detected. An IDS should not be deployed without a Concept of Operations (CONOPS) and a set of well-defined goals, host profile characteristics, and responses and tuning approaches.

Often, tuning requires evaluating IDS operation for a period of time at initial activation (some implementations do self-tuning) and then tuning out or desensitizing the monitor. Sometimes sensitivity may need to be increased, but most technologies come out of the box highly sensitive. Anomaly detection elements usually have a learning curve to determine normal patterns and distributions of activity. Finally, the adjustments can be made to deselect some activities and add others based on the analysis and correlation of alarms and alerts with other measures in the system.

Response Options

Although the sensors collect information about intrusions, it is the analyst who interprets the results. Host-based IDS agents watch aspects of host or server security, such as OS log files, access log files, and application log files, as well as user-defined application policies. If a policy is breached, the host IDS can react by logging the action, alerting the administrator (notify a console, send e-mail, beep a pager), disabling an account, terminating an intruder’s session, shutting the system down, or executing a command that in some cases stops the action before execution.

Reporting Mechanisms

When the host IDS determines that the criteria have been met for declaring an intrusion, anomaly, or misuse event, it is generally configured to signal alerts to either a console interface or a centralized management station where information can be brought to the attention of an administrator. Some host IDSs can send e-mails, from the central console or individual agents, to alert an operator to events or initiate telephone pages if properly configured.

As with network IDs, many host-based IDs, central-reporting systems come with database components that allow the general manipulation or correlation of event data, as well as the generation of a wide variety of reports, both graphical and numerical.

7.2.1.1 General Considerations for Selecting the Technology

Rather than scanning network packets, a host-IDS watches the audit logs or other system resource events and activities on each monitored computer for signs of misuse. Host-based IDSs are easy to configure for individual servers and applications. They provide tailored security because they can monitor specific OS or application events and can enforce enterprise policies. Only host-based IDSs can detect an intrusion that occurs through the locally attached console, and when an attack is detected, only these IDSs can enforce a user-based reaction policy (e.g., disable the user account or terminate a user process).

A host IDS is well suited to monitoring specific user and file activity. However, because it cannot detect network-based threats, a host-based IDS should be considered a complement to network-based IDSs, supplementing detection of intrusions that may appear to be part of authorized traffic flows or that might otherwise be missed within switched environments. While use of both technologies is preferred, there are situations where it may be appropriate to use host-based IDS only—

- Network bandwidth is too high to enable network monitoring, or too low to justify the expense of a network IDS.
- The network environment is highly switched (logically segmented), without span ports on the switches, or the mesh is too large, making the number of sensors needed prohibitively expensive.
- The topology is too highly distributed (either geographically or logically segmented).
- Organizational/domain communities of interest or ownership issues (e.g., different organizations own the network and the hosts or a subset of the hosts, and these organizations do not communicate well).
- There are privacy or consent issues; it is much easier to have a “consent to monitor” policy when logging into a host than a network.

A classic case in which host-based IDSs are the only practical approach is a high-performance computing community where a loose coalition of high-end computing environments shares data, but the owners of the processing capacity do not own the network.

Host-based IDS performance varies according to the number of standard attack definitions and enterprise-specific policies being monitored, and the number and type (compute-bound versus input/output-bound) of processes executing on the host, as well as the speed of the host and its components. Another factor is the enterprise architecture for host management.

Although intrusion detection and response systems are important components of an enterprise security program, the devices currently in use have many flaws. Host-based IDSs rely on after-the-fact analysis of audit data to detect suspicious activity and anomalies and are difficult to scale for use in large enterprise environments. In addition, they may cause computational overhead on mission-critical servers and hosts whose security is being monitored, because the IDS resides on the same machine.

Another consideration is complexity of deployment and administration, which varies depending on how many and what types of servers are being protected. A host-based IDS cannot address attacks that exploit protocol vulnerabilities, and since IDSs analyze data from the audit trails, they typically do not react to an attempted intrusion in real time. Moreover, the access to audit trails is available only at the OS or the application level; that is why host-based IDSs should be implemented in the context of a total Defense-in-Depth security posture with a comprehensive approach to enclave boundary security.

Table 7.2-1 summarizes the advantages and disadvantages of host-based IDS technologies.

Table 7.2-1. Host-Based IDS Considerations

Advantages	Disadvantages
Provides a real-time measure of the adequacy of a system's access control and protection. Systems can monitor who accessed the system. Systems can map problem activities to a specific user ID. Systems can track behavioral changes associated with information system misuse, typical of an insider of the information system. Systems can operate in an encrypted environment. Systems can operate in a switched network environment. On large networks, systems can distribute the load associated with monitoring across available hosts.	Network activity is not visible to host-based sensors. False alarm rates are high with current technologies. Activating audit mechanisms can add to resource overhead on the system. Audit trails used as data resources can take up significant storage space. Operating system vulnerabilities can undermine the integrity of host-based sensors and analyzers. The management and deployment costs for host-based systems are greater than for other approaches to intrusion detection system. Host-based sensors are more platform-specific, which adds to their cost and the expertise required of operators.

Finally, the degree to which the host-based IDS is configured to monitor a particular system should depend on the sensitivity of the information being processed or the criticality of the system to the integrity and availability of the entire enterprise.

Host-based IDS systems come with operational and managerial burdens. These include alerts that require specific administrator examination, implementations that may be available only for specific OSs, and system performance that affects the host. Without careful planning, a broad deployment of host-based IDSs is not recommended. A threat and risk assessment is strongly recommended to identify particular hosts on which to add IDSs, followed by a careful deployment and continual monitoring for performance impact or operational degradation.

7.2.1.1 Important Features

In selecting host-based IDS, a number of features should be considered. This section identifies those features, and the next section discusses the rationale for choosing the features.

Detection

- Support for detection of service start-up.
- Ability to detect registry changes.
- Ability to watch files and objects.
- Ability to profile normal activities and detect variations from the norm.

Signatures

- The number of events and signatures that can be detected.
- Checking for file or message integrity that is based on cryptographic algorithms, not simple checksums.
- Customizable system checks.

Operations

- Deployment and management capabilities of the complete IDS system (e.g., number of agents that can be connected to a single manager and number of managers that can report to a single console).
- Ability of the auditing process to automatically reset itself.
- Support for remote management.
- Ability to integrate with network-based modules; how well the tool works in a heterogeneous environment becomes a critical factor for enterprise-class IDS tools.
- Survivability characteristics (self-recovery from power loss, resource failure, component failure, and similar situations).

Response Options

- Configurable, automated, rule-based response capabilities.
- Account blocking, access control changes.
- Ability to coordinate responses across multiple host platforms (e.g., disable the same account on all enterprise systems).
- Integrated response with network-based tools.

Reporting Options

- Ability to perform Simple Network Management Protocol (SNMP or trap) alerting to centralized system management.
- Ability to use e-mail alerts and a variety of other contact measures (pager, fax, etc.) to notify personnel.
- Ability to execute programmed scripts automatically on alerts at management system or console (also partially a response function).
- Ability to generate customized reports as needed.
- Ability to capture events in a standardized database system.

Performance

- Balance between the overhead required to audit OS and application activity logs and the ability to react to infractions.
- Effect of data log on system resources (since host-based IDS generates log files as well).

Platform

- The specific types of platforms (e.g., OS) on which the tool operates.
- Minimum platform configuration.
- Memory requirements.
- Disk resource requirements.
- Ability to handle crossover when reporting between platforms.

Console Considerations

- **Operator Interface—**Command and monitoring provisions available to an operator.
- **Mark as Analyzed—**Ability to clear or mark alarms that have been reviewed.
- **Drill Down—**Ability to provide additional information for selected events.
- **Correlation—**Tools to correlate events based on source, destination, and type.
- **Report Generation—**Ability to generate reports upon event detection and as periodic summaries.
- **Integrated Industry—**Standard database.

7.2.1.3 Rationale for Selecting Features

In choosing detect-and-respond capabilities, operations and personnel considerations must be integrated into the technology solutions, consistent with the overall Defense-in-Depth philosophy. Because host-based monitoring does not itself offer protection from intrusions or attacks, it should be considered more as instrumentation that monitors and measures the effectiveness of the host computer's existing protection structures. It is up to system administrators (support and operations staff) to interpret IDS outputs and reports and initiate the response. If full-time operators[1] are not available to interpret IDS outputs and formulate responses, IDS implementations will typically not add real value and IDS deployments should probably not be considered.

If an IDS is being considered, a number of factors must be taken into account based on how the IDS is intended to be used, whether full- or part-time operators will be available, and how skilled the operators are in interpreting the results.

Detection

Most host-based IDS technologies actually use a mix of both signature matching and anomaly or misuse detection. Both have advantages. Although signature-based IDSs are traditional, they typically cannot detect new or modified attack patterns. While many intrusions, particularly by novices, use standard attack sequences (often downloaded from hacker bulletin boards), an accomplished adversary will be able to create new attacks or modify old attacks and thus thwart traditional signature detection mechanisms.

Anomaly and misuse detection approaches (e.g., statistical profiling and unauthorized system resource use or modification monitoring) have greater flexibility for identifying new or modified attacks because they monitor network usage or behavior. These are also the only mechanisms currently available to monitor actions of otherwise authorized users for misuse, whether inadvertent or intentional. They can sometimes be more complex to operate and manage, but in most technologies, the degree to which each aspect (signature versus misuse/anomaly) is enabled and configurable.

As always, any decision is based on level of risk, anticipated performance, cost (for purchase, deployment, and operation), and operational impact. This framework recommends deployment of multiple attack detection schemes, where possible, to increase the likelihood of detection.

1 Ideally operators should be available round the clock every day. The number of operators needed will depend on the traffic loads and the likely numbers of incidents. Hundreds of thousands of intrusion alerts per day are not uncommon, and each has to be investigated to determine whether the threat is serious.

Signatures

In a signature-based IDS or IDS component, it is desirable to have as many signatures as possible available. However, increasing the size of the signature set will decrease the overall performance of most IDSs. Since the lists of possible attacks change often, it is strongly recommended that the IDS be capable of dynamically loading signatures. It is usually operationally more feasible and efficient if the downloading is handled on an enterprise (or at least site) basis. Most vendors that offer dynamic loading of signatures provide periodic updates to the signature base; a good rule of thumb is that having more frequent updates is better. If operators have the skills to create custom signatures, the ability to support user-defined attacks is also desirable, particularly if custom attacks are found at a site.

Operations

Easy configuration of the IDS according to the security policies of the information system being monitored is desirable. The IDS should also be able to adapt to changes in system and user behavior over time (e.g., new applications, users changing from one activity to another, or new resources that cause changes in system resource usage patterns).

By their nature, IDS sensors are located where intrusions are likely. IDS sensors are also high value targets in themselves. To this end, if such modifications occur, an IDS component within a host system should be self-monitoring, detecting unauthorized modifications and notifying an attendant console. To simplify return of full operations after an intrusion, it is also desirable that the IDS be able to recover from system crashes, either accidental or caused by malicious activity, and be able to recover its previous state upon start-up.

Response Options

Many solutions offer automated response options that seem on the surface to be very desirable. They imply the need for little or no human interaction, as the devices can provide an immediate response. Unfortunately, though, it is not uncommon for a host IDS, depending on where it is employed, to identify as potential misuse many events that are in fact characteristic of normal host usage. Without careful tuning, the number of false positives may be high, giving rise to unwarranted indications of intrusions. Automated responses that terminate user sessions, modify access controls, throttle processes, or actually shut down a system can often cause severe denial-of-service threats to the network. It is strongly recommended that automated options not be used unless there is some mechanism to control the potential for denial of service.

Reporting Options

Most host-based IDSs report alarms to an operator console (see the discussion of console features below). Which level and frequency of reporting are desirable depends primarily on the skills of the operators available. Some host IDS technologies offer the option of paging or sending e-mail messages to notify personnel of alarms. While these sound desirable, they may give rise to operational issues: With an IDS detecting thousands of alarms a day, these features might overload e-mail servers, creating a denial-of-service threat themselves, or page operators far too often at all times of the day and night. These features are generally not recommended, at least not until a baseline of normal behavior is identified.

Performance

Host IDS performance varies based on the available resources (processor, bus architecture, disk space) of the host system, the operational applications it is executing, the number and type of processes it experiences during operations, the number of attack signatures employed, and the level and complexity of audit or analysis the IDS is configured to undertake. Unlike network-based intrusion detection sensors, where performance degradation results in the loss of intrusion detection capabilities but not network performance, host-based sensor software can affect the entire host system itself. In each case, a trade-off must be determined between the levels of audit the sensor software is configured to undertake and the effect on overall system performance. Where existing host performance is already marginal, redesign of the system and sensor software deployment approaches should be considered—host-based IDSs must be deployed very carefully.

Platform

A major issue in selecting host-based IDS is the type of computer skills (e.g., UNIX, NT) required of operators. They are likely to need the skills necessary to install, configure, adjust, and maintain the system. Since a host-based IDS is usually deployed in an existing system, knowing what is already running on the system and the resources it requires is critical. In addition, the console platform must be acquired and maintained, so it is useful to select a technology that functions on the platforms used within the enterprise.

Console Considerations

As discussed in Section 8.2, Detect and Respond as a Supporting Element, the primary function of the console is to help characterize and analyze the many alarms that will be identified. Operators must identify alarms that resulted from authorized use (e.g., false alarms), those that do not offer serious risks to the network, and those that do; they must also gain an initial perspective on the source and impact of possible attacks.

Operator Interface—The type of interface that is operationally desirable tends to depend on operator preference. Novices typically prefer a graphical user interface (GUI) with intuitive operations, pull-down screens, and substantial aids. More skilled operators may prefer command string operations, tailored screen options, and more customization options. It is best if operators can get a hands-on trial of console capabilities before final selection.

Mark as Analyzed—Because operators will typically be faced with large numbers of alarms to be analyzed and cleared, the ability to keep track of alarms that have been reviewed is usually critical.

Drill Down—Many host IDS consoles display a high-level characterization of events in order to display the large number of alarms that are detected. Operators must usually access additional details about each alarm to characterize it properly. It is very desirable that the console be able to provide these additional levels of information upon request. As with the operator interface, the types of information desired depend on the skills of the operators.

Correlation—As with drill-down features, operators need tools for correlating incidents (e.g., based on source, destination, type of alarm or event) to identify and properly characterize intrusions and attacks, particularly when the incidents are distributed in time or location. Console ability to integrate the reporting of host-based and network-based IDSs and other relevant events is a strong plus—if the operators will use the additional information. Again, as with the operator interface, the types of tools that will be useful typically depend on the skills and mission of the operators.

Reporting—The reporting options will depend predominantly on the type of information operators want for characterizing intrusions and the organization's need for reporting to higher levels (e.g., periodic summary reports). It is always desirable for the console to be able to generate reports that can be disseminated with little extra operator effort.

Considerations for Deployment

A host-based IDS is designed to monitor a single host on which it (or its agent) resides. Typically, it can watch data available from higher levels of protocol stacks, which restricts its ability to monitor activities to audit trails made by the OS or applications. It also can detect the activities that occur locally on the monitored host (e.g., file permission modification and user account setup).

Host-based IDSs fall into two basic configurations: single system and agent/manager. A single system IDS protects one machine by detecting intrusions in the machine's audit logs and through other methodologies. A manager/agent host-based IDS places agents on one, some, or all hosts; IDS agents reside on the systems that are to be monitored. These host-based systems rely on analysis of OS event logs and audit processes (among other techniques described above) to detect suspicious activity. They are part of a distributed architecture in which the system agents report to a centralized management station, with agents connected to managers that are connected to a central console. Agents can remotely upgrade or install new versions and attack-signature rules. This configuration allows security administrators to define and distribute rules from one central location.

Some host monitors can also track audit trails from other applications, like firewalls, Web servers, and routers. These fall into the category of network-based monitoring capabilities, which are discussed in Section 6.4, Network Monitoring Within Enclave Boundaries and External Connections. While the Information Assurance Technical Framework (IATF) focuses on the technology aspects of an overall IA solution, the value of an IDS is realized only when a competent operator or analyst can interpret the result. Operators must be trained to ensure that they have the analytical skills and proficiency with tools to make correct interpretations efficiently. They also need procedures (e.g., courses of action, standard operating procedures) for all contingencies, particularly for when serious attacks are discovered.

7.2.1.4 Considerations for Operation

Most IDS technologies can tune the sensor to improve its performance for specific deployments. When an IDS is first deployed, it is prudent to complete tuning by operating the technology for some time (depending on the complexity of the deployment). This provides an opportunity for determining that the IDS can monitor applications and detect alarms and for increasing or decreasing sensitivities. Also, anomaly detection elements usually have a learning curve for establishing a baseline for normal patterns and distributions of activity. The tuning period also allows for other adjustments to deselect some activities and add others based on an analysis of the alarms triggered.

Tuning enables the IDS to preclude detection of authorized traffic patterns that may otherwise cause false-positive alarms. There are two fundamental approaches to tuning. The first is to have prior knowledge of the usage patterns that could trigger false alarms. The IDS can then be configured (tuned) to preclude these from causing an alarm. Unfortunately, it is often not possible to have this information in advance. The other approach is to run the IDS and to have it find conditions that generate alarms. As alarms are detected, an analyst determines whether there was an actual intrusion, or whether the alarm was the result of a false positive based on normal operation. The IDS can then be tuned to preclude those events from triggering an alarm. This method also gives operators an opportunity to become familiar with the technology before it becomes operational.

Tuning should not be thought of as strictly an installation process. It should be performed regularly to refine and focus the detection mechanisms on real intrusions and reduce false positives.

Once an IDS is deployed, it is recommended that the IDS be tested to ensure that it is configured correctly and is functioning properly. While it is also possible to construct exercises to test the proficiency of the operators and analysts, normal day-to-day operations are likely to provide more than enough real alarms to provide opportunities to assess their capabilities.

7.2.2 Host Monitors—Malicious Code or Virus Detectors

Over the past decade, computer viruses[2] have gone from an academic curiosity to a persistent, worldwide problem. Viruses can be written for, and spread on, virtually any computing

2 The term "virus" is often misused as referring to *anything* that "infects" a computer *and* causes damage. A more appropriate term for any software that attacks a system is "malicious code." Nevertheless, in the following paragraphs, the term *virus* encompasses all malicious code and delivery mechanisms.

platform. When first introduced, they were often structured as boot sector attacks, typically promulgated by first infecting the floppy disks that are read during start-up. Because the primary file transfer mechanisms today are electronic means such as e-mail, boot sector viruses are no longer a major concern. Typically, viruses today are written to affect personal computers (PC); if the PC is connected to other machines on a local area network (LAN), it is possible for the virus to invade these machines as well. Section 6.6, Malicious Code Protection, contains detailed descriptions of various types of malicious code, potential malicious code attacks and countermeasures, and requirements for detecting malicious code.

7.2.2.1 Technology Overview

Malicious code scanning technologies prevent or remove most types of malicious code. Using these technologies with current virus definitions is crucial in preventing and detecting all types of malicious code attacks.

There are several basic categories of antivirus technologies:

- **Preinfection Prevention Products—**A first line of defense against malicious code, used before a system has been attacked.
- **Infection Prevention Products—**Used to stop replication processes and prevent malicious code from infecting the system.
- **Short-Term Infection Detection Products—**Used to detect an infection very soon after it has occurred.
- **Long-Term Infection Detection Products—**Used to identify specific malicious code on a system that has been infected for some time, usually removing the malicious code and returning the system to its prior functionality.

Section 6.6.5.2, Viruses and E-Mail, contains a more detailed description of malicious code detection technologies.

7.2.2.2 General Considerations for Selecting the Technology

Workstations with individual access to networks or information service should have malicious code protection, as should networks at the gateway (see Section 6.4.2, Malicious Code or Virus Detectors). Malicious code can destroy data through network connections if allowed past the gateway or through individual user workstations. Although a single user can bring an infected disk to work, infecting his or her workstation and eventually the entire network, most malicious code infections result from file sharing. Because so many individual users now keep all data files on networks or shared file systems instead of diskettes, continuous protection of network connections at the gateway is important.

7.2.2.3 Important Features

In selecting antivirus technologies, a number of features that should be considered. These technologies are identified in this section, and the rationale for selecting them is discussed in the next section. Additional factors to consider when selecting a malicious code detection product can be found in Section 6.6.6, Selection Criteria.

Detection Capabilities

- Data integrity checks.
- Ability to exploit malicious mobile code.
- Real-time virus scanning.
- On-demand virus scanning.
- Recognition of—
  - Different strains of polymorphic viruses.
  - Viruses residing in encrypted messages and compressed files.
  - Viruses in different languages (e.g., JAVA, ActiveX, Visual Basic).
  - Trojan horses and worms.

Updates

- Ability to upgrade an existing version.
- Availability of regular updates.
- Frequency of update releases.

Response Mechanisms

- Quarantine at the server level.
- Quarantine at the console level.
- Network-based responders.
- Alerts sent to network or system administrators.
- Alerts (in the case of e-mail-borne viruses) sent to sender and all receivers.

Platform Considerations

- What platforms the tool runs on.
- Availability of cross-platform support.

7.2.2.4 Rationale for Selecting Features

When selecting antivirus technologies, two important guidelines should be followed. The “best” technology may not be good enough by itself. Also, since data security technologies operate in different ways, one technology may be more useful than another in different situations. Keeping these guidelines in mind and rating each of the following categories will allow an organization to choose the best malicious code protection technology for its unique needs.

Detection Capabilities

Most computer-virus scanners use pattern-matching algorithms that can scan for many different signatures at the same time (see Section 6.6.5.2, Viruses and E-Mail). Malicious code detection technologies must be able to detect known and unknown worms and Trojan horses. Most antivirus technologies search hard disks for viruses, detect and remove any that are found, and have an auto-update feature that enables the program to download profiles of new viruses so that it can scan for them. The virus signatures these programs recognize are quite short—typically 16 to 30 bytes out of the several thousand that make up a complete virus—it is more efficient to recognize a small fragment than to verify the presence of an entire virus, and a single signature may be common to many different viruses.

Although antivirus applications are essential for the detection of known viruses, no mail filter or malicious code scanner can defend against a new mail worm attack. Although the recent Love Bug virus was caught quickly, it still did a wealth of damage, and it is only a matter of time before crackers figure out how to send e-mail worms that infect systems without attachments having to be opened.

Updates

Defending against virus and hostile-code threats takes far more than the ability to produce perfect detection rates at a single point in time. With an average of nearly 300 new viruses discovered each month, the actual detection rate of antivirus software can decline rapidly if the program is not kept current. As new viruses are discovered, so are corresponding cures to update protections. Antivirus systems should perform these updates automatically, reliably, and through a centrally controlled management framework. This is why an enterprise-class antivirus solution must be able to offer timely and efficient upgrades and updates across all client and server platforms.

Response Mechanisms

Once malicious code has been detected, it must be removed. One technique is simply to erase the infected program, but this is a harsh method of elimination. Most antivirus programs attempt to repair infected files rather than destroy them. A virus-specific scanning program that detects an infected file can usually follow a detailed prescription, supplied by its programmers, for deleting the virus code and reassembling a working copy of the original.

There are generic techniques that also work well for both known and unknown viruses. One method is to gather a mathematical fingerprint for each program on the system so that if a program is later infected, a copy of the original can be reconstituted. Most tools scan for viruses, but not all detect and remove Trojan horses, worms, and malicious mobile code at all levels of entry. Most current antivirus tools do not have the same capabilities when responding across a network. (Additional countermeasures related to malicious code can be found in Section 6.6.4, Potential Countermeasures.)

The technology should be easy to use, with clear and uncluttered menu systems and meaningful screen messages. Help systems are essential, as users need current information on all types of malicious code. The trend is to provide online help; however, the technology should come with manuals. The malicious code protection technology should be compatible with all other software and hardware, and create no conflicts. The company that produces the technology should be stable and able to provide local technical support for all questions and problems. The technology should be fully documented. All messages and error codes should be deciphered, and full installation guides and how-to manuals should be provided.

Platform Considerations

The computers to run this software must meet the hardware and software requirements specified by the manufacturer. Malicious code protection software should perform its duties without failing itself or interfering with other applications running on the same system.

7.2.2.5 Considerations for Deployment

Defense in Depth dictates that any virus protection must be implemented across the enterprise, on every system. Although some advocate only installing antivirus protection only on edge devices, such as servers, firewalls, and gateways, defense against viruses is only as good as its weakest link. If one system can be compromised, the entire enterprise is at risk.

Centralized antivirus management that imposes common policies is strongly recommended. Though some vendor offerings make end users responsible for security mandates, this can lead to more and more varied security holes. What often happens is that users have a session interrupted with a pop-up screen says their files are about to be scanned or they are about to receive an antivirus update. Many users then override the update manually, because it is distracting.

7.2.2.6 Considerations for Operation

Most antivirus technologies send responses or alerts at the server level, and some at the console level. It is always desirable to notify anyone whose files may have been infected that malicious code has been detected, especially system and network administrators. When malicious code is encountered in e-mail transactions, it is desirable to notify both sender and recipient. If it is found on a file system that knows the file owner, that person should be notified. In general, anyone who could be notified should be.

7.2.3 Host Vulnerability Scanners

In addition to the on-line host monitoring technologies that provide a critical layer of defense within enclave boundaries, another class of technologies—host scanners—can also be deployed to improve overall security. The distinction between these scanners and network monitoring devices is that monitors typically operate in near real time and tend to measure the effectiveness of the host's protection services. This is more an "after the fact" measure than a preventive measure. Scanners, on the other hand, are preventive measures. Typically, they operate periodically (or on demand), examining hosts for vulnerabilities that an adversary could exploit. They measure security effectiveness.

Scanning can be performed at two levels. A remote (or network) scanner is run over a network against the target node, probing it for vulnerabilities. Here the software is running on an administrative system and scanning a target anywhere on the network (see Section 6.5, Network Scanners Within Enclave Boundaries). A local (or host) scanner runs as a software program that resides on the node itself. Host scanners are discussed here.

Unlike near-real-time host monitoring technologies, host scanners are typically executed periodically or on demand, providing perspectives on the posture of a local environment. Section 8.2, Detect and Respond as a Supporting Element, provides a perspective on an overall detect and response infrastructure, but because these assessments typically look at the local level, they tend not to interact with or be particularly relevant to a broader system infrastructure.

7.2.3.1 Technology Overview

Host-based vulnerability scanner tools examine the security posture of a host system from within, unlike network-based tools, which scan from the viewpoint of the network. Host scanners examine the contents of files looking for configuration problems, comparing what they find with predefined policies or best practices, and generating alerts when they detect possible security deficiencies. These technologies catch security problems that are not visible at the network level and that could be exploited by users with malicious intent who already have access to the system through valid means (or otherwise, such as stolen authentication information).

Detection

Scanners compare data about the host's configurations with a database of known vulnerabilities. They work either by examining attributes of objects (e.g., owners and permissions for files) or by emulating an attacker. In the latter approach, they run a variety of scripts to exploit any vulnerabilities in the host. Most scanners can be configured to select which vulnerabilities to scan for and when. Some scanners allow operators to incorporate their own scanning routines to look for site-specific application weaknesses. Some also offer capabilities for grouping hosts and customized options by scan group.

Scan Configuration Mechanisms

Each host in an enclave should be equipped with a host-based scanner. If the number of nodes is small, locally configuring the scanner and reviewing the results may be preferred in order to minimize network traffic overhead. If the network is large, it is often desirable to configure one or more consoles to control distributed node scanners. Some technologies have software distribution frameworks for propagating this control. Hosts can be collected into groups, and a host can be a member of more than one group. Groups can be scanned at different times, with variations in the vulnerabilities inspected tailored to each group, enabling the operator to scan some hosts "deeper" than others. For example, one can configure the scanners to search for user configuration errors on hosts that serve many users and omit those scans on hosts (e.g., servers) that have no users.

Response

When a host is scanned, some technologies create a "fix script" recommending corrective actions. It may be possible to customize this script or to run it to eliminate the vulnerabilities identified. Some also provide an unfix script that lets operators undo the fix script.

7.2.3.2 General Considerations for Selecting the Technology

One advantage of periodic scanning is that resource utilization is less on average than that required for real-time monitoring, because processing resources are required only when the scanner is active. Unlike host monitoring technologies that are intended to catch adversaries in the act, scanners reveal weaknesses that could be exploited later. Since host scanners actually run on the target node, they can look for problems that cannot be detected by remote (network) scans. They can also inspect patches to ensure that the latest security fixes have been installed. The obverse is that because scanners are run only periodically, they do not detect malicious events as they occur.

7.2.3.3 Important Features

In selecting host-based vulnerability scanners, a number of features should be considered. This section identifies these features; the next section discusses the rationale for choosing them.

Scanning Capabilities

- Ability to add custom scanning routines to look for site- or technology-specific weaknesses.

Signature/Vulnerability Database

- Comprehensive list made of vulnerabilities in the target host.
- Periodic updates from the vendor.
- Ease of adding entries by the user.
- Database backed by a vendor-funded research center, rather than just culled from Internet-based sources of vulnerability information or some combination of the two.

Response Mechanisms

- Vulnerable ports of entry automatically shut off.

User Interfaces

- Reports viewable in real time.

Reporting Capabilities

- Automatic alerting when new nonnetwork ports are detected.
- All system answers logged in a database or file.
- Updated database of network numbers with which to compare newly identified numbers.
- Automatic combination of information logged into database, organized in a report format.
- Ability to suggest mitigation approaches for vulnerabilities discovered.

Platform Compatibility

- Platforms (OS) on which the tool will run.
- Use of executables.
- Support for scripts or macros.

Source

- For tools developed by the Government (or under Government sponsorship) information on whether tool is reserved and whether your organization can get authorization for its use.
- Reputation of the vendor.
- Availability of source code for tools in the public domain (e.g., freeware from the Internet).

7.2.3.4 Rationale for Selecting Features

The type and level of detail of information tools provide varies greatly. Although some can identify only a minimal set of vulnerabilities, others perform much more analysis and provide detailed recommended mitigation approaches. Select scanner technologies that cover the gamut of vulnerabilities for the given OS (e.g., UNIX or Windows), including password vulnerabilities, access control, resource and file permission signatures, registry problems, and the like. Select also technologies that offer a comprehensive library of vulnerabilities periodically updated by the vendor. For larger environments, capabilities like grouping of nodes into scan groups and customized scan options may be valuable.

Some scanner technologies offer features whose usefulness depends on the training and skills of their operators. Depending on planned usage and operator skills, it is often desirable to select technologies that can be tuned to ignore some false positives. It is also desirable to select features that enable the scanner to be tuned for specific application environments, such as databases, Web servers, file servers, and firewalls, because such profiles may differ for the function the system must provide to the enterprise.

Signature/Vulnerability Database

A significant characteristic of host-based vulnerabilities is that they tend to be unique to an OS, and even an application. Some applications that are portable also port their vulnerabilities across platforms, and can have different vulnerabilities on different platforms. And, obviously, operating structures differ drastically between the general UNIX base (and its variants), Windows 95/98, and Windows NT/2000. It is therefore important that the vulnerability database provided for the host-based IDS be comprehensive, adaptable, and well maintained by the vendor. IATF strongly recommends selecting technologies from vendors that do their own research and have specific expertise in OS vulnerabilities, rather than those that simply incorporate vulnerability signatures culled from other Internet-based resources.

Response Mechanisms

Assessment tools will continue to evolve, with some vendors offering click-and-fix solutions. Assessment software flags vulnerabilities in terms of the risk posed to the network and the ease of the fix. Some technologies can generate trouble tickets to trigger a manual response. They may make it possible to change policies in firewalls and other enclave boundary defense mechanisms. Some identify patches that should be installed. Some offer to obtain and install patches. Although installing patches is feasible, security administrators can do it; in fact, the difficulty of undoing configuration changes makes this feature less desirable. Consider such features in light of the environment's current configuration management policies and procedures.

User Interfaces

Typically, scanners are already configured with lists of vulnerabilities and can operate without customization. Some technologies allow operators to customize the vulnerabilities the scanner will investigate, and when. Newer tools provide user-friendly front ends and sophisticated reporting.

Reporting Capabilities

Usually scan results are sorted into a file that can be accessed on demand. Old technologies inundated customers with phonebook-sized reports on all the vulnerabilities that the network faced. New technologies have database interfaces that prioritize vulnerabilities, allowing network managers to deal with problems logically. Many generate reports that are Web-enabled, with hot-links and other "labor savers." For sites with only a few platforms, running the scans and reading the reports on each node may be appropriate. For sites with large numbers of hosts, it might be wise to consolidate reports at a central server. If this feature is selected, it is recommended that technologies chosen offer encryption for information transferred from the local hosts to the centralized server to protect the scan results information.

Platform Compatibility

The computers to run this software must meet the hardware and software requirements specified by the manufacturer. Vulnerability scanner software should perform its duties properly without inadvertently causing any of the monitored systems to fail or bringing anything else down. Technologies chosen should therefore have minimal effect on the performance of the host, and provide for cooperative computing resources for other services and applications on the host.

Source

Host vulnerability scanner technologies are available from a variety of sources. Various Government organizations have created their own tools, usually for use by specific communities. The quality of these tools varies according to the skills and the testing of the developing organization. Use of any of these is likely to require authorization.

Other tools are available commercially or can be downloaded over the Internet. Unless Government tools have been found to be effective, commercial tools available from reputable vendors are recommended. Download from the Internet only if the source code is available so that the tool can be evaluated by an experienced analyst. If source code is not available, the tool may not detect actual vulnerabilities and worse, might actually introduce vulnerabilities (e.g., as a source of a malicious code attack).

7.2.3.5 Considerations for Deployment

It is often useful to deploy vulnerability scanners in conjunction with a host-based IDS. An IDS will be able to identify when a file has been modified; however, it cannot determine what changes were made to that file. If there is a scanner, the IDS can invoke it to inspect the contents of the file. Maintaining configurations of owners, groups, and permissions for files and directories is one typically challenging task; scanners can ensure that these aspects of a security policy are properly implemented.

7.2.3.6 Considerations for Operation

It is important to specify when, as well as what, scans are performed. Otherwise, mission-critical servers might become busy responding to simulated attacks during times of peak demand.

Assessment frequency is a function of how often network changes are made as well as enterprise security policy. Depending on the organization, assessments may take place quarterly, monthly, weekly, or even daily. Some service providers offer subscription scanning services, ensuring that assessments take place regularly.

It is recommended that features that provide automated vulnerability repair be disabled. If they are not, the system must be backed up fully (including all system and application software) before any automated repair.

7.2.4 File Integrity Checkers

File integrity checkers are a specialized type of host scanner that verify the integrity of the files, detecting when files have been changed. As with the host vulnerability scanner technologies already discussed, these technologies tend to run off-line and thus are not a protection mechanism. Typically they operate periodically, based on an event (e.g., file access), or on demand.

7.2.4.1 Technology Overview

This is a small, tailored class of technologies, configured with the location of specific key configuration files or executables (depending on the OS in question) that are typically targeted by attackers attempting to compromise the system. These might include the registry environment, file permissions, security policy, and account information. The software typically generates cryptographic checksums of the targets and periodically probes to see whether the files have been surreptitiously modified. The best known of these technologies is Tripwire, but there have been some more recent entries into the field. A few host-based IDS monitors and vulnerability scanners have limited file integrity checking capabilities, and a number of technologies that started out as integrity checkers are evolving into policy violations checkers and vulnerability scanners. In fact, the two product lines are coalescing.

Most integrity checkers use the same general paradigm. They operate on files identified from a library of known files to monitor. Depending on the platform and OS, the technology creates unique identifiers typically based on cryptographic checksums, then stores them for future use. When the file integrity program is executed, either automatically or manually, new unique identifiers are calculated. The integrity checker compares the new identifiers with the saved versions and notifies the operator or administrator when a mismatch shows that the file has been modified or deleted. The operator or administrator determines whether the differences result from intrusive activity.

7.2.4.2 General Considerations for Selecting the Technology

General considerations for use of file integrity checkers closely parallel those of host IDS and vulnerability scanning in general, with a few additional discriminators. Most important is that file integrity checkers are supported by cryptography, providing stronger protection against their defeat by intruders. File integrity checkers that are configured to run in near real time provide instantaneous indication of attack or failure, and if they are configured to run on files or data structures that do not change, their alarms require little or no interpretation.

Unfortunately, file checkers suffer from the same performance and resource consumption drawbacks as other host-based technologies. It is also critical to ensure that the baseline signatures from which the checkers function are both well protected from modification and, if they are dynamic configuration data structures, are created before the system is accessible to users. Table 7.2-2 summarizes the advantages and disadvantages of file integrity checkers.

Table 7.2-2. File Integrity Checker Considerations

Advantages	Disadvantages
Checkers use cryptographic methods to provide additional security protections. Checker gives clear immediate evidence of intrusion when files that should never be modified are discovered modified, unlike host-based IDS reports, which must be interpreted, and alarms, which must be intercepted. System can operate within an encrypted environment because the host has access to decrypted versions of files. On large networks systems can distribute the load associated with monitoring across available hosts.	Network activity is not visible to host-based sensors. Checker may cause additional resource overhead on the system, depending on frequency of execution. OS vulnerabilities can undermine the integrity of host-based sensors and analyzers. File identifiers or signatures, even if based on cryptographic checksums, must have their own strong protection. Management and deployment costs of host-based systems are often greater than in other IDSs. Host-based sensors are often platform-specific, which adds cost and requires more operator expertise. If not deployed before system is operational, checker may miss early system compromises.

7.2.4.3 Important Features

In selecting host-based file integrity checking scanner, a number of features should be considered. This section identifies these features; the next section discusses the rationale for choosing them.

Scanning Capabilities

- Monitor each OS with comprehensive files and data structures (including data structure and directories environments, such as Lightweight Directory Access Protocol [LDAP] or full X.500 services).
- Strong cryptographic checksums implemented as part of the identifier scheme.
- Centralized reporting for large enterprises.
- Built-in analysis or recommended action when modification is noticed.
- Self-checking.

- Easy specification of additional files/structures to monitor.

Response Mechanisms

- Automated restoration of “clean” file or data structures.

User Interfaces

- GUI for number entry, dialing status, and call results.
- Reports be viewable in real time.

Reporting Capabilities

- Automatic alert when new, nonnetwork ports are detected.
- System answers logged in a database or file.
- Updated database of network numbers with which to compare newly identified numbers.
- Automatic combining of logged information into a report format.
- Provision of suggested mitigation approaches for discovered vulnerabilities.

Platform Compatibility

- Platforms (OS) on which the tool will run.
- Use of executables.
- Support for scripts or macros.

7.2.4.4 Rationale for Selecting Scanning Features

We strongly recommend technologies that offer a comprehensive library of files and data structures for tracking that is periodically updated by the vendor. As new vulnerabilities are discovered that include files or structures that an attacker might modify, vendors should provide immediate updates.

Strong cryptography should be implemented as part of the checksum creation and recheck. Most scripted attack programs already compensate for widely known simple checksum hashing techniques and recalculate checksums. Additionally, some integrity checking technologies can now monitor static portions of directory structures, such as those found in LDAP or full X.500 directory environments.

As with host vulnerabilities, file and data structures integral to any particular OS tend to be unique to an OS or even an application. Some applications that are portable also port their vulnerabilities across platforms, and can have different vulnerabilities (characterized by different targeted files or data structures) on different platforms. And, obviously, operating structures differ drastically between the general UNIX base (and its variants), Windows 95/98, and Windows NT/2000. It is therefore critically important that the database of files and data structures to monitor that is provided for the host-based integrity checker be comprehensive, adaptable, and well maintained by the vendor. We strongly recommend selecting technologies from vendors that do their own research and have specific expertise in OS vulnerabilities, rather than those that simply incorporate vulnerabilities signatures culled from other Internet-based resources.

Response Mechanisms

Assessment tools will continue to evolve, with some vendors offering click-and-fix solutions. This will be true in the file integrity-checking environment as well, with some tools able to restore, from a secured backup environment, files or environments that have been illegally modified.

User Interfaces

Most file checkers enable the operator to configure which files and data structures are monitored and when, although typically the checkers are preconfigured with lists of files and data structures to watch and can operate without customization. Newer tools have user-friendly front ends and sophisticated reporting capabilities.

Reporting Capabilities

Usually file integrity check results are sorted into a file that can be accessed on demand. Old technologies inundated customers with phonebook-sized reports on all the vulnerabilities the network faced. New technologies have database interfaces that prioritize vulnerabilities, allowing network managers to deal with problems in a logical manner. Many generate reports that are Web-enabled, with hot-links and other labor savers. For sites with only a few platforms, running the checks and reading the reports on each node may be appropriate. For sites with large numbers of hosts, it might be wise to consolidate reports on a central server. If this feature is selected, it is recommended that technologies chosen offer encryption for information transferred from local hosts to the centralized server to protect the file integrity check information.

Platform Compatibility

The computers to run this software must meet the hardware and software requirements specified by the manufacturer. File integrity checking software should perform its duties properly without failing and with minimal effect on the performance of the host.

7.2.4.5 Considerations for Deployment

The decision on whether and how to deploy these programs includes understanding how often to run the integrity recheck step, whether it should be done automatically or by operator command, and where the reports are to be centralized. These all depend on the sensitivity of the information being processed and how critical that system is to the rest of the enterprise.

7.2.4.6 Considerations for Operation

The most important question is the timing of deployment. To be most effective, integrity checkers should be initialized before systems are placed in production and made generally accessible to their user communities. If files and data structures are baseline-monitored any time after a system has "gone live," the system may already be compromised and the integrity checker will miss changes that have already occurred. This is particularly true in structures that are not supposed to remain static (e.g., access control databases, unlike static executables that should not change from their installed release).

7.2.5 Typical Bundling of Capabilities Within Products

At one point, host monitors were offered as stand-alone devices. A number of offerings now combine these monitors with firewalls, routers, vulnerability scanners, and the like, as vendors try to leverage existing market positions to gain market share in related areas. Another emerging trend is for larger vendors to offer integrated architecture approaches, bundling a number of related technologies. Vendors tend to prefer custom rather than standard interfaces to preclude the merging of other vendor offerings. These "complete solutions," however, tend to lock the buyer into a single product suite. While this may sound attractive, it is often more valuable to be able to integrate various technologies to take advantage of the detection capabilities of different implementations.

There is a natural linkage of these monitoring technologies with enterprise security management (ESM) systems. For several years, it has been expected that host-based vulnerability assessment software will be integrated into system management platforms and that aspects of network-based products will find homes in network management platforms, but there is little evidence that this will happen in the immediate future.

7.2.6 Beyond Technology Solutions

While the focus of this IATF is on technology solutions, there are important operational aspects of effective network monitoring that are critical to an effective IA solution.

Operational Planning

We recommend the following:

- Build intrusion detection and antivirus activity into the enterprise security policy.
- Assess the ability of system administration personnel to perform intrusion detection and vulnerability scanning.

- Consult with experienced intrusion detection and vulnerability scanning personnel about the best approach.
- Seek a balanced and symbiotic deployment of sensors.
- Consult with legal counsel about the rights and procedures of affected personnel (see below).
- Provide adequate technical and legal training of all affected personnel.
- Acquire software and expertise from a vendor of known integrity.
- Monitor networks consistent with enterprise security policy.
- Tightly couple vulnerability scanning and intrusion detection.
- In detecting intrusions—
  - Look for intrusion evidence based on found vulnerabilities; use intrusion evidence to find and correct vulnerabilities
  - Provide and monitor bogus sites, services, and information. Monitoring intrusions through known vulnerabilities may satisfy the prosecution requirements of legal authorities
  - Use intrusion responses that are approved by the appropriate authority
- In detecting network malicious code attacks—
  - Select and deploy virus scanning that is consistent with location, functions, and capabilities.
  - Acquire or download antivirus software from a high-integrity source and acquire any necessary hardware (e.g., an ancillary firewall that scans incoming or outgoing traffic for viruses).
- Institute enterprise wide antivirus procedures and training.
- Scan consistently based on time or events.
- Follow up on all indications of potential contamination (as defined in the enterprise security policy and antivirus procedures).
- Update antivirus software and hardware as needed (e.g., consistent with new releases of antiviral software and specific experiences throughout the enterprise).

General Activities

- Archive (within any legal constraints) audit and intrusion information and correlate with vulnerability scan information.
- Keep authorities apprised of all activities, so that no legal rights are violated.
- Continuously repeat steps, as appropriate.

Privacy Concerns

Organizations may own the intellectual property created by employees and may also legally restrict computer activities to those approved by management. A common practice is to warn all users of this as part of the normal login message.

This does not mean that all managers own all the transactions of all the employees. Especially unclear is how to handle the conflict between privacy and necessary monitoring. Use of IDSs and system-monitoring tools requires caution. Sniffers that search for key words in messages (such as “attack,” “weakness,” or “confidentiality”) as standard watchwords may find them used in an appropriate manner depending on the type of correspondence. Audit trail reports may contain full command strings (including parameters). Knowing that an employee is sending several messages to a particular department (e.g., Human Resources) may infringe on his or her privacy. It is important to refer privacy concerns to the legal and policy parts of the enterprise before technologies are deployed and used.

7.2.7 For More Information

The reference materials used in preparing this section (listed at the end of the section) provide an excellent base of knowledge on relevant technologies; there are also a number of other sources of information. This section deals primarily with on-line sources because they tend to offer up-to-date information.

7.2.7.1 IATF Executive Summaries

An important segment of the IATF is a series of executive summaries that offer implementation guidance for specific situations. These offer important perspectives on the realistic operation of specific technologies. As these are formulated, they will be posted on the IATF Web site http://www.iatf.net. [1]

7.2.7.2 Protection Profiles

National Security Telecommunications and Information Systems Security Policy (NSTISSP) No. 11 sets out the national policy that governing the acquisition of IA and IA-enabled IT products for national security telecommunications and information systems. Effective January 2001, preference was to be given to products that comply with one of the following:

- International Common Criteria for Information Security Technology Evaluation Mutual Recognition Arrangement.
- National Security Agency (NSA)/National Institute of Standards and Technology (NIST) National Information Assurance Partnership (NIAP).
- NIST Federal Information Processing Standard (FIPS) validation program.

Since January 2002, this requirement is mandated. Department of Defense (DoD) Chief Information Officer (CIO) Guidance and Policy Memorandum No. 6-8510, Guidance and Policy for Department of Defense Global Information Grid Information Assurance incorporates NSTISSP No. 11 as an acquisition policy for the DoD.

The International Common Criteria and NIAP initiatives base product evaluations on Common Criteria protection profiles. NSA and NIST are working on a comprehensive set of protection profiles. An overview of these initiatives, copies of the protection profiles, and status of various products that have been evaluated are available at the NIST Web site, http://niap.nist.gov/.[2]

7.2.7.3 Independent Third Party Reviewers of Technologies

ICSA Net Security Page, www.icsa.net.

Talisker's Intrusion Detection Systems, www.networkintrusion.co.uk/.

Network Computing–The Technology Solution Center, www.nwc.com/1023/1023f12.html.

Paper on CMDS Enterprise 4.02, http://www.Intrusion.com/Products/enterprise.shtml (ODS Networks has changed its name to Intrusion.com).

Paper on CMDS Enterprise 4.02, http://www.Intrusion.com/Products/enterprise.shtml (ODS Networks has changed its name to Intrusion.com).

PC Week On-Line, www.zdnet.com/pcweek/reviews/0810/10sec.html.

7.2.7.4 Relevant Research Sites

Coast Homepage–Purdue University, www.cs.purdue.edu/coast.

University of California (UC) Davis, http://seclab.cs.ucdavis.edu/

7.2.7.5 Selected Host Monitor and Scanner Vendors

Axent Technologies, www.axent.com.

cai.net, http://www.cai.net/.

Cisco Connection Online, www.cisco.com.

CyberSafe Corporation, www.cybersafe.com.

Internet Security Systems, www.iss.net.

Network ICE, www.networkice.com.

References

1. Information Assurance Technical Framework (IATF), http://www.iatf.net.

2. National Institute of Standards and Technology, http://niap.nist.gov/.

Additional References

a. Amoroso, Edward, Intrusion Detection. Intrusion.Net Books, 1999.

b. AXENT Technologies, Inc. Intruder Alert 3.5 IDS Review Guide, May 2000.

c. AXENT Technologies, Inc. *Everything You Need to Know About Intrusion Detection*, 1999.

d. Balasubramaniyan, J. S., et al. An Architecture for Intrusion Detection Using Autonomous Agents. COAST Technical Report. 11 June 1998.

e. Concurrent Technologies Corporation. Attack Sensing, Warning, and Response (ASW&R) Baseline Tool Assessment Task Anti-Virus Trade Study Report. Report No. 0017-UU-TE-000623. 13 April 2000.

f. Concurrent Technologies Corporation. Attack Sensing, Warning, and Response (ASW&R) Trade Study Report Intrusion Detection System. Report No. 0017-UU-TE-000621. 14 April 2000.

q. Department of Defense (DoD) Chief Information Officer (CIO) Guidance and Policy Memorandum No. 6-8510, Guidance and Policy for Department of Defense Global Information Grid Information Assurance.

h. Escamilla, Terry. Intrusion Detection, Network Security Beyond the Firewall. Wiley Computer Publishing, 1998.

i. Graham, Robert. “New Security Trends for Open Networks.” SC Magazine, October 1999.

j. Information Assurance Technology Analysis Center (IATAC). Tools Report on Intrusion Detection. Defense Technical Information Center. December 1999.

k. Information Assurance Technology Analysis Center (IATAC). Tools Report on Vulnerability Analysis Information. Defense Technical Information Center. 15 March 2000.

l. Maes, V. “How I Chose an IDS.” Information Security Magazine. Volume 2, Number 9. September 1999.

m. National Security Telecommunications and Information Systems Security Policy (NSTISSP) No. 11. National Policy Governing the Acquisition of Information Assurance (IA) and IA-Enabled Information Technology (IT) Products. January 2000.

n. Northcutt, Stephen. Network Intrusion Detection, An Analyst’s Handbook. New Riders Publishing, 1999.

o. SC Magazine. “Intrusion Detection.” June 2000.

p. Schneider, Sondra, et al. “Life After IDS.” Information Security Magazine. Volume 2, Number 9, September 1999.

q. Snapp, Steven R., et al. A System for Distributed Intrusion Detection. IEEE CH2961-1/91/0000/0170, 1999.

r. Ulsch, MacDonnell, and Joseph Judge. “Bitter-Suite Security.” Information Security Magazine. 2, # 1. January 1999.

This page intentionally left blank